VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

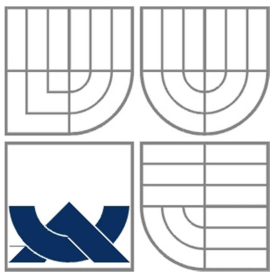
EXTRAKCE INFORMACÍ Z WIKIPEDIE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

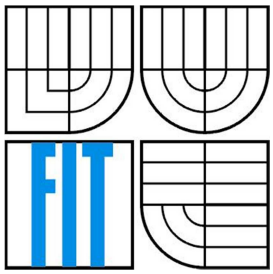
AUTOR PRÁCE
AUTHOR

MARTIN MUSIL

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

EXTRAKCE INFORMACÍ Z WIKIPEDIE

INFORMATION EXTRACTION FROM WIKIPEDIA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN MUSIL
AUTHOR

VEDOUCÍ PRÁCE — Ing. MAREK SCHMIDT
SUPERVISOR

BRNO 2011

Abstrakt

Tato bakalářská práce se zabývá tématem automatické extrakce informací z textu. Cílem je vytvoření aplikace, která za užití extrakčních vzorů získává znalosti z článků informačního internetového serveru Wikipedie. V úvodu jsou vysvětleny základní pojmy této problematiky, hlavní část práce se věnuje experimentům a především samotné implementaci rozdělené do dvou částí – zpracování textu a následného získávání informací. Vyhodnocením projektu je pak samotná analýza výsledků experimentů a efektivita vytvořených pravidel.

Abstract

This bachelor thesis deals with the problem of automatic information extraction from text. Goal is to create an application, which captures knowledge out of the articles from online information server Wikipedia, using extraction patterns. At the beginning, we interpret the basic terms of the subject and the main part of the publication is focused to the experiments and above all to the implementation, divided into two parts, processing of the text and following information extraction. The conclusion of the thesis analyses the results coming from experiments and efficiency of created rules.

Klíčová slova

Wikipedie, automatická extrakce informací, pravidlový systém, Python

Keywords

Wikipedia, automatic information extraction, rule-based system, Python

Citace

Martin Musil: Extrakce informací z Wikipedie, bakalářská práce, Brno, FIT VUT v Brně, 2011

Extrakce informací z Wikipedie

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Marka Schmidta
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Musil
17.5.2011

Poděkování

Velice rád bych poděkoval svému vedoucímu Ing. Markovi Schmidtovi za pomoc, trpělivost, ochotu a čas strávený při konzultacích k této práci.

Obsah

Obsah 1
1 Úvod 3
2 Stav poznání 4
2.1 Wikipedie 4
2.2 Extrakce informací 5
2.2.1 Aplikační užití 6
2.2.2 Typy získaných struktur 7
2.2.3 Typy nestrukturovaných zdrojů 8
2.2.4 Vstupní zdroje pro extrakci 8
2.2.5 Metody extrakce 9
2.2.6 Výstupy extrakčních systémů 9
2.2.7 Požadavky na extrakci 9
2.3 Pravidlové metody 10
2.3.1 Forma a zastoupení pravidel 10
2.3.2 Vlastnosti tokenů 10
2.3.3 Pravidla 11
2.3.4 Organizace sbírky pravidel 12
2.3.5 Pravidla učící se algoritmy 12
2.3.6 Shrnutí 13
2.4 JAPE 13
2.5 Reprezentace znalostí 14
3 Návrh 15
3.1 Základní problémy 15
3.2 Použité nástroje 15
3.3 Návrh a definice jazyka 16
3.4 Rychlost vs. pokrytí 16
3.5 Konečný automat 17
4 Implementace programu 18
4.1 Prostředí 18
4.2 Chování aplikace 18
4.3 Části skriptu 18
4.3.1 Ovládání aplikace 19
4.3.2 Předzpracování dat 19
4.3.3 Datové struktury 20

4.3.4 Počáteční analýza objektů 20
4.3.5 Problematika zpracování a stavby vět 21
4.3.6 Problematika zájmen a nepřímé řeči 23
4.3.7 Částečná kontrola validity věty 23
4.3.8 Hlavní extrakční analyzátor entit 23
5 Testování pravidel 27
5.1 Vstupní data 27
5.2 Zkoumané oblasti 27
5.2.1 Extrakce informací o městech 28
5.2.2 Extrakce dat o politicky unifikovaných oblastech 31
5.3 Korektnost úprav při analýze souvětí 34
5.4 Diskuse k problémům a nedostatkům 36
5.5 Vyhodnocení extrakce 36
6 Závěr 38
6.1 Zhodnocení práce 38
6.2 Možná rozšíření 38
Literatura 39
Seznam příloh 40
Obsah přiloženého DVD 41

1 Úvod

Způsoby šíření informací se postupně vyvíjely od prostých gestikulací přes řeč a tištěné slovo až po nástroj současnosti v podobě všepřítomného internetu, díky kterému jsme schopni nalézt hledaná data bez vynaloženého fyzického úsilí během několika vteřin. Jedním z hlavních důvodu, proč je dnes otázka dohledání faktů tak jednoduchá, je existence tzv. webových informačních portálů, obsahujících rozsáhlé databáze dat. Jedním z takových serverů je i systém Wikipedie, který díky jedinečným vlastnostem tvoří a vyvíjejí uživatelé internetu sami. Témata jsou v něm popsána a vysvětlena pomocí vyčerpávajících článků, tento způsob prezentace faktů je však mnohdy nepohodlný jak z pohledu hledání a selekce informací pro lidské poznání, tak z pohledu problematiky extrakce dat automatizovaných strojů vytvářejících znalostní databáze.

Naším úkolem je tedy navrhnout a implementovat systém, který by za užití extrakčních metod z databáze článků Wikipedie, a tedy v offline režimu, vyhledal o jednotlivých tématech stručný výtah základních informací.

V druhé kapitole jsou vypsány a vysvětleny základní pojmy, se kterými se pro řešení problému je třeba seznámit. Obsahuje hlavní informace o principu wiki a serveru Wikipedie, stručný výtah o problematice automatické extrakce informací z nestrukturovaných textů, případy jejího využití a techniky, které se při ní užívají.

Třetí část popisuje princip, prvotní koncept aplikace a návrh systému. Dále je v něm obsažen výčet externích pomocných aplikací, které budou pro správný chod systému využity, odstavec o námi vytvořeném jazyce užitém pro značkování větných prvků, konečný automat pravidel a zamyšlení nad možnými problémy, se kterými se při implementaci můžeme setkat.

Kapitola čtvrtá se pak zabývá samotnou implementací aplikace. Popisujeme v ní jednotlivé klíčové funkce programu a další důležitá témata s nimi spojená. Stěžejní body tvoří popis zpracování vstupního textu a extrakční část skriptu. Pro lehčí porozumění problematiky jsou často k teorii uváděny příklady.

V předposlední kapitole se nachází text, obsahující zadání experimentů a charakteristiku zkoumaných pravidel. Následně je popsán samotný průběh testování a analýza výsledných znalostí, díky které pak určujeme procentuální přesnost užitých předpisů, a tedy i přesnost extrakce.

Závěrečná kapitola shrnuje a hodnotí úspěšnost dosavadní práce, stav poznání a problémy se kterými jsme se v průběhu projektu setkali. Poslední krátký odstavec diskutuje možnosti rozšíření a dalšího teoretického využití aplikace.

2 Stav poznání

2.1 Wikipedie

Obrázek 2.1: Logo Wikipedie

S touto ikonou se můžeme na internetu setkat poměrně často, ať již jako odkaz na zdroj informací v článcích na informačních serverech, v podobě různých wiki pluginů pro mobilní či desktopové aplikace sloužících pro co nejrychlejší vyhledání dat bez nutnosti užití internetového prohlížeče, či při jiných příležitostech.

Pod zkratkou ‘wiki’ si všichni IT uživatelé představí ten největší a nejnavštěvovanější informační server na celosvětové internetové síti, avšak tato problematika je daleko složitější. Samotné slovo ‘wiki’ pochází z Havaje, kde značí autobusovou letištní dopravu a v překladu znamená ‘rychle’. Tento výraz od roku 1995, kdy byl poprvé užit pro software sloužící ke psaní a diskusi o vzorových jazycích, v počítačové sféře symbolizuje soubor internetových informačních portálů, které díky své otevřenosti umožňují uživateli nejen vyhledávat články a čerpat z nich informace, ale především je modifikovat a vytvářet texty nové. Ke konci 20. století se potenciál wiki systémů stával čím dál více populárním pro návrhy soukromých i veřejných znalostních bází. Jednou z nich byla také samotná Wikipedie, která spatřila světlo světa v roce 2001. Původně byla založena na *UseMode*[1] systému a až v průběhu svého počátečního vývoje přešla na vlastní open source kód, který se později stal vzorem pro mnohé další wiki systémy [1]. Jako první vznikla Wikipedie anglická, která obsahuje v současnosti nejrozsáhlejší databázi článků ze všech wiki portálů. Dále následovala její německá verze a postupně se tento systém rozšířil celosvětově a v současné době má kompletní databáze Wikipedie přes 15 milionu článků, psaných ve 240 jazycích. Díky své všestrannosti a globálnosti se pak toto číslo každým dnem zvětšuje.

Ve tradičních wiki systémech jsou články reprezentovány třemi způsoby: HTML kódem, výsledným zobrazením tohoto kódu prohlížečem a uživatelsky editovatelným zdrojovým kódem [1]. Posledně zmiňovaný formát je známý jako tzv. ‘*wikitext*‘[2] a je psán ve zjednodušeném značkovacím jazyce, jehož styl a syntaxe může být v jednotlivých implementacích odlišná. Důvodem tohoto systému je fakt, že HTML formát je díky své velké zásobě vnořených značek komplikovaným

[1] Původně užívaný systémem Wikipedie, psaný v jazyce PERL.
[2] Značkovací jazyk užívaný pro online tvorbu článků Wikipedie.

a pro běžného uživatele internetu složitým jazykem. Tvorba či editace článků by byla obtížným, zdlouhavým procesem a syntaxe by odváděla pozornost od samotného obsahu stránky. Pro příklad uvedeme názornou ukázku všech tří způsobů prezentace.

Zobrazený text	Syntaxe v MediaWiki[3]	HTML kód
„*The Danube* is *Europe's* second longest river after *the Volga*“	„''The Danube'' is ''Europe's '' second longest river after ''the Volga '' ''	„ The Danube is Europe's second longest river after the Volga “

Tabulka 2.1: Způsoby reprezentace dat

2.2 Extrakce informací

Automatická extrakce informací z nestrukturovaných zdrojů vytváří čistou sémantiku strukturovaných databází, a tím otevírá nové cesty pro dotazování, organizování a analyzování dat. Kořeny extrakce se nacházejí v komunitě přirozeného zpracování jazyka, kde primární impuls přichází se snahou rozpoznání jmenných entit, jako například jmen osob v titulcích novinových článků. V posledních letech se společnost, mající snadný přístup ke zdrojům strukturovaných a nestrukturovaných dat, stává stále více informačně závislou, což dalo za vznik novým použitím těchto metod. Současný zájem společnosti spočívá v převodu našich osobních počítačů ve strukturované databáze, v přeměně znalostí z vědeckých publikací ve strukturované záznamy a využití internetu pro dotazování nad těmito informacemi. V důsledku tohoto vývoje existuje mnoho vědeckých týmů výzkumných pracovníků, vytvářejících techniky od strojového učení, databází, vyhledávačů informací a počítačové lingvistiky, až po různé aspekty problematiky extrakce informací [2].

Extrakce informace se vztahuje k automatické extrakci strukturovaných dat, jako jsou entity, vztahy mezi entitami a atributy je popisující z nestrukturovaných zdrojů. Tento fakt umožňuje mnohem širší formy dotazů nad bohatými nestrukturovanými zdroji, než by bylo možné za užití vyhledávání pouze podle klíčového slova.

Extrakce struktur z hlučných a nestrukturovaných zdrojů je velice složitým úkolem, kterým se zaobírá široká komunita výzkumníků po více jak dvě desetiletí. Tímto tématem, majícím kořeny ve společenství zpracování přirozeného jazyka (NLP[4]), se nyní zabývá velké množství odvětví, zahrnující strojové učení, shromažďování informací, databáze, web či analýzy dokumentů. Prvotní extrakční úlohy byly soustředěny kolem identifikace jmenných entit v přirozeném jazyku, jako jsou lidé či názvy společností a vztahy mezi nimi. Oblast působnosti tohoto výzkumu byla silně ovlivněna dvěma vývojovými programy, Asociací chápání zpráv (dále již jako MUC[5]) a Automatickou extrakcí obsahu (dále jen ACE[6]). Nástup internetu značně zvýšil rozsah a různorodost aplikací, závislých na určitých formách extrakce informace. Online programy, jako stránky porovnávající nákupy a další

[3] Wiki PHP systém na kterém je postavena Wikipedie.
[4] Věda zabývající se interakcí mezi počítači a přirozeným jazykem
[5] Message Understanding Conference: výzkum technik zpracování přirozeného jazyka.
[6] Automatic Content Extraction: výzkum technic automatické extrakce obsahu.

automatické portálové aplikace, vedly k šílenství ve výzkumu a obchodních aktivitách v tomto sektoru [2].

Pro splnění požadavků a potřeb těchto různorodých aplikací se techniky strukturální extrakce v průběhu posledních dvou desetiletí postupně vyvíjely. První systémy byly tzv. pravidlové, založené na ručně psaných pravidlech. Toto jejich manuální vytváření se stalo postupem času více a více pracnější a nedostačující, proto byly vyvinuty algoritmy pro jejich automatické učení. V další etapě byly extrakční systémy užívány na čím dál hlučnější nestrukturované zdroje a pravidla přestala být efektivní. Tak přišel na řadu věk systémů statistického učení. Jak se zvětšoval rozsah extrakčních systémů, tato metoda nabízela ucelenější analýzu struktur dokumentů. Navzdory postupné evoluci různých technik není v této soutěži žádný vítěz a poražený. Obě metody, pravidlové a statistické, se nadále v současné době používají bok po boku a výhody té či oné varianty závisí na charakteru extrakčních úkolů. Existují i hybridní modely, které se snaží vzít z obou výše zmiňovaných variant to nejlepší [2] a [3].

2.2.1 Aplikační užití

Strukturální extrakce jsou užitečné v celé škále aplikací. Dále uvedeme jejich reprezentativní podmnožiny, zařazené buď jako aplikace firemní, osobní, vědecké či web-orientované, a ty nejdůležitější si rozebereme.

Firemní aplikace: V tomto odvětví se extrakce užívají například ke sledování zpráv. Zde to jsou klasické aplikace extrakce informací, automaticky sledující konkrétní typy událostí ze zpravodajských zdrojů. Populární MUC a ACE soutěže jsou založeny na získávání strukturovaných entit jako jména osob či firem a vztahů mezi nimi, pro názornost „*is-CEO-of*“. Jiné časté užití spočívá ve sledování ohnisek nákazy či teroristických akcí ze zpravodajských zdrojů. Jako další stojí za zmínku péče o zákazníky. Klientsky-orientované firmy shromažďují mnoho forem nestrukturovaných dat z interakce se zákazníky, jako jsou jejich osobní údaje či produkty z objednávek. Tato data jsou většinou přímo propojena s vlastními strukturovanými databázemi. Společnost se tak díky těmto znalostem může přizpůsobovat a měnit obchodní strategie.

Správa osobních informací: Systémy správy osobních informací (PIM) se snaží ve strukturované vnitřně-spojené podobě organizovat osobní data, jako jsou dokumenty, e-maily, projekty či personální údaje. Úspěch těchto systémů závisí na schopnosti automaticky extrahovat strukturu z existujících, převážně souborových, nestrukturovaných zdrojů.

Web orientované aplikace: Druhů těchto aplikací existuje celá řada. Citační databáze byly vytvořeny přes komplikované fáze extrakcí struktur ze všech možných zdrojů, počínaje konferenčními weby a jednotlivými domovskými stránkami konče. Populárními nástroji jsou Citeseer[7], Google Scholar3[8] nebo Cora[9]. Následně existuje nespočet webových stránek plnících funkci názorových databází, ukládajících nemoderované stanoviska na řadu témat včetně produktů, knih, filmů, lidí a hudby. Hodnota těchto stanovisek může být výrazně zvýšena, pokud jsou organizovaná do strukturovaných polí, například pro produkty by mohlo být užitečné zjistit převládající polaritu názorů na každý rys produktu. Dalším příkladem vytváření strukturovaných

[7] Digitální knihovna bibliografických informací.
[8] Služba pro ucelené vyhledání odborné literatury.
[9] Systém analyzující hlavičky a citace v textech.

databází z webových dokumentů jsou společenské weby jako DBLife[10] a Rexa[11], které sledují informace o výzkumech, konferencích, rozhovorech, projektech a událostech týkajících se konkrétních komunit. Vytvoření takových strukturovaných databází vyžaduje mnoho extrakčních kroků, jako například zachycování textových prohlášení, získávání jmen řečníků, titulků a podobné. Dalším odvětvím je srovnávací nakupování. V současnosti je na trhu velký zájem o vytváření webových serverů, které automaticky prochází stránky obchodníků a vyhledávající produkty, jejich parametry a ceny, které pak mohou být použity pro porovnání. Poslední a asi nejužívanější kategorií jsou strukturované webové vyhledávače. Složitá problematika extrakce informací umožňuje uvádět na internetu strukturované vyhledávací dotazy týkající se subjektů a jejich vztahů. Vyhledávání podle klíčového slova jsou vhodná pro získávání informací o entitách, kterými jsou typicky podstatná jména nebo jmenné fráze. Tato technika selhává při potřebě užití dotazů, hledajících vztahy mezi subjekty. Například pokud bychom chtěli vyhledat data, obsahují text ve formě *„Společnost X získala Společnost Y“*. Vývoj prototypů pro zodpovězení těchto druhů dotazů neustále běží.

2.2.2 Typy získaných struktur

Získané informace kategorizujeme do čtyř typů: entity, vztahy mezi entitami, adjektiva popisující entity a výše řazené struktury, jako jsou tabulky či seznamy.

Entity jsou typicky jmenné fráze, skládající se z jednoho či více tokenů v nestrukturovaném textu. Nejvíce populární formu tvoří jmenné entity, jako jsou jména osob, míst a společností. Rozpoznání jmenných entit bylo poprvé představeno v šesté verzi MUC a sestávalo ze tří dílčích úkolů: hledání vlastního jména a zkratky osob, míst, organizací (ENAMEX), absolutní časové podmínky (TIMEX) a peněžního či jiného číselného vyjádření (NUMEX). Nyní je škála jmenných entit značně rozšířena, aby zahrnovala i generika, jako jména nemocí, tiskovin a podobných termínů. ACE soutěž pro extrakci vztahů entit z textů přírodního jazyka jich uvádí více než 100 různých typů.

Vztahy jsou definovány nad dvěma a více entitami, které se nacházejí na předem definovaných pozicích. Příkladem je *„is employee of”*, zastupující vazbu mezi osobou a organizací, *„Is acquired by”* jako vztah mezi dvojicemi společností, *„location of outbreak”*, charakterizující spojitost mezi onemocněním a místem či *„is price of*“, což je vztah mezi názvem výrobku a finanční částku. Extrakce vztahů se od extrakcí entit v jednom ohledu významně liší – i když entity odkazují na sekvenci slov ve zdrojovém textu a můžou být vyjádřeny jako anotace nad zdrojem, vztahy však nejsou anotacemi na podmnožině slov. Místo toho vyjadřují asociace mezi dvěma samostatnými textovými úryvky reprezentujícími subjekty. Výpis vícecestných vztahů se často odkazuje na záznamovou extrakci. Populárním druhem tohoto systému je extrakce události. Například pro událost jako je vypuknutí epidemie, extrahujeme vícecestné vztahy zahrnující název nemoci, místo ohniska nákazy, počet lidí postižených či mrtvých a datum vypuknutí. Některé záznamové extrakce jsou triviálnější, protože nestrukturovaný řetězec znamená pevnou sadu vztahů. Například pro adresy, kde je vztah *„is situated in“* hledán mezi extrahovaným názvem ulice a města. Další forma vícecestných vztahů v přírodních jazykových společenstvích je sémantické značení role, kde je cílem k danému predikátu, ve větě identifikovat jeho různé sémantické argumenty.

[10] http://dblife.cs.wisc.edu/
[11] http://rexa.info/

Přídavná jména popisující entity jsou v mnoha aplikacích potřebná, hodnota těchto adjektiv typicky musí být odvozena kombinací měkkých klíčů, rozšířených do slov v okolí entity.

Rozsah extrakčních systémů je nyní rozšířen o těžbu nejen atomických entit a plochých záznamů, ale také bohatších struktur, jako jsou ***tabulky, seznamy a stromy*** z různých typů dokumentů.

2.2.3 Typy nestrukturovaných zdrojů

Rozřazujeme je do dvou dimenzí: Základní jednotka zrnitosti, na které je spuštěn extraktor a různorodost ve stylu a formátu, v nestrukturovaných dokumentech.

Zrnitost extrakce: Nejvíce populární forma extrakce je z malých textových úryvků, jako jsou buď nestrukturované záznamy – například adresy, citace a inzeráty, nebo věty extrahované z odstavců přirozeného jazyka. V případě nestrukturovaných záznamů mohou být data považována za soubor strukturovaných spojených polí, případně polí s omezeným přetvářením. V případě mnoha složitějších extrakčních úkolů je pro smysluplnou extrakci nezbytné zvážit souvislosti mezi několika sousedními větami nebo celým dokumentem. Populární příklady zahrnují extrakce událostí ze zpravodajských článků, problémů z e-mailů ve střediscích pomoci, extrakce z hlaviček a citace z vědeckých publikací. Techniky navržené v tomto problému se většinou zaměřují na první část zdrojového textu. Pro získávání informací z delších jednotek je typicky hlavním úkolem navrhnutí efektivních postupů pro filtrování pouze relevantní části dlouhého dokumentu. V současné době je tato problematika zvládnutelná přes ručně psanou heuristiku.

Různorodost nestrukturovaných zdrojů: Důležitým faktorem, majícím obrovský dopad na složitost a přesnost extrakce je informace o tom, jak moc jsou nestrukturované texty homogenní. Kategorizujeme je jako:

- *Strojově generované stránky:* Populárním zdrojem v tomto prostoru jsou HTML dokumenty, dynamicky generované prostřednictvím databází.
- *Částečně strukturované domény specifických zdrojů:* Zdroje užívají daný neformální styl, který je užíván v celém jejich rozsahu, a tak je možné vyvinout rozumný extrakční model. Například novinové články, klasifikované reklamy nebo citace.
- *Otevřené zdroje:* V poslední době je zájem extrahování instancí vztahů a entit z otevřených domén jako je web. V této problematice je důležitým faktorem využití redundance vytěžených informací napříč mnoha různými zdroji.

2.2.4 Vstupní zdroje pro extrakci

Základní specifikace otázky extrakce zahrnuje jen ty druhy struktur, které budou čerpány, a nestrukturované zdroje, z nichž by měly být extrahovány. V praxi existuje několik dalších vstupních zdrojů, které jsou k dispozici na podporu extrakce.

- *Strukturované databáze* známých entit a vztahů jsou cenným zdrojem pro zlepšení přesnosti extrakce.
- Výrazně cennějšími zdroji než strukturované databáze jsou *označené nestrukturované zdroje*, protože nabízí kontextové informace o entitě, a také proto že forma, v níž se

jednotka v nestrukturovaných datech nachází, je často velmi hlučná forma jejího výskytu v databázi.

- Mnoho extrakčních systémů podstatně závisí na *předzpracovaných knihovnách*, které děj obohacují jazykovými nebo uspořádávajícími informacemi, sloužícími jako cenné body pro uznávání struktur.

2.2.5 Metody extrakce

Metody pro extrakci informací kategorizujeme do dvou rovin: ručně-kódované nebo učící se a pravidlové či statistické.

Ručně-kódované a učící se systémy si detailněji probereme v následující kapitole. *Pravidlové metody* extrakce jsou řízeny tvrdými predikáty, jsou snadněji interpretovatelné a rozšiřitelné, naproti tomu *metody statistické* mají rozhodování založené na váženém součtu predikátových střel a jsou účinnější na hluk v nestrukturovaných datech. Z těchto informací vyplývá, že pravidlové systémy mají využití v uzavřených doménách, kde je nezbytný a dostupný lidský faktor. V otevřených doménách jako je extrakce faktů z přepisů řečí či extrakce názorů z blogů je pro změnu vhodnější měkká logika statistických metod [2] a [3].

2.2.6 Výstupy extrakčních systémů

Existují dva základní režimy, v nichž se nasazují extrakční systémy. První variantou je ten, ve kterém potřebujeme identifikovat všechny zmínky o strukturované informaci v nestrukturovaném textu, a za druhé tehdy, pokud je cílem naplnění databází strukturovaných entit. V tomto případě se koncový uživatel po vytažení entit o vstupní nestrukturovaný text dále nestará. Jádra většiny technik zůstávají stejná bez ohledu na formu výstupu, jen v několika druzích otevřených extrakcí, kde se užívá redundance pro zlepšení spolehlivosti konečných dat uložených v databázích, jsou podstatnější rozdíly.

2.2.7 Požadavky na extrakci

Rozsáhlé implementace extrakčních modelů vyvolávají mnoho různých požadavků, dvěma hlavními jsou správnost a výkon, dalšími méně důležitými například udržovatelnost či použitelnost.

Největší výzvou pro vědeckou obec je navrhování modelů, které dosahují vysoké *správnosti extrakce*. Některé faktory, přispívající k obtížnosti dosažení vysoké přesnosti při těžbě úkolů jsou rozmanitost klíčů, obtížnost nalezení chybějících extrakcí a zvětšená složitost extrahovaných struktur. Pojem správnost se v oboru extrakci informací skládá ze dvou pojmů: přesnosti a pokrytí.

- *Přesnost* charakterizuje procentuální poměr získaných entit, které byly extrahovány ve správném formátu a obsahují požadovanou znalost.
- *Pokrytí* pak vypovídá údaj o poměru všech informací, obsažených ve vstupním textu a znalostech, které byly nalezeny.

Nasazení extrakčních technik v reálném světě přináší několik praktických *výkonnostních problémů*. Za prvé chceme mechanismy účinně filtrující správnou podmnožinu dokumentů, která by měla obsahovat požadované strukturované informace, za druhé potřebujeme prostředky pro efektivní

přiblížení části dokumentu, obsahující důležité informace, a konečně se musíme zabývat mnoha drahými kroky zpracování, kterými vybraná část textu může projít [2].

2.3 Pravidlové metody

Mnoho extrakčních úloh lze řešit pomocí kolekce pravidel, která jsou buď ručně psaná, nebo učící se z příkladů. Prvotní systémy získávání informací byly všechny pravidlové a v současné době i nadále pokračuje jejich vývoj a úpravy k řešení problémů extrakčních systémů. Pravidla jsou zvláště užitečná a výkonná, když je úloha řízená a má určitou standardní syntaxi, jako například extrakce telefonního čísla a PSČ z e-mailů, nebo když vytváříme obálky pro strojově generované webové stránky. Pravidlové systémy jsou oproti statistickým také rychlejší a snadněji se optimalizují. Typicky se takový systém skládá ze dvou částí: sbírek pravidel a mechanizmů pro řízení střelby v případě že jsou pravidla vícenásobná.

2.3.1 Forma a zastoupení pravidel

Pravidlové systémy mají dlouhou historii užívání a v průběhu let se vyvinulo mnoho různých formátů jejich reprezentace. Například společný šablonový specifikační jazyk (CSPL) a jeho deriváty jako JAPE[12], vzory a seznamy položek v Rapieru[13], regulární výrazy ve WHISKu[14], SQL výrazy v Avataru a datalogové výrazy v DBLife. Pravidla popisujeme obecným způsobem zachycujícím základní funkce většiny z těchto jazyků.

Základní pravidlo je ve tvaru: Kontextový vzor → Akce. Kontextový vzor se skládá z jednoho nebo více označených vzorů, zachycujících různé vlastnosti jedné nebo více entit, a kontextu, ve kterém se objevují. Značené pravidlo se skládá ze vzoru, který je většinou regulárním výrazem, definovaným nad rysy tokenů v textu a volitelným štítkem. Ten je používán pro odkazování se na odpovídající tokeny v pravidlové akci.

Akční část pravidla se užívá k označení různých částí značkování: přiřazení entitního štítku k posloupnosti tokenů, vkládání začátku nebo konce entitní značky na pozici nebo přidělení více entitních značek.

Většina pravidlových systémů je kaskádovitá, pravidla jsou aplikována ve více fázích, z nichž každá přiřazuje vstupnímu dokumentu anotaci, která slouží jako vstupní charakteristika do další fáze. Například extraktor kontaktních adres lidí je vytvořen ze dvou fází pravidlových anotátorů: první fáze označuje tokeny s entitními popiskami jako lidé, jména, emailové adresy či geografická umístění obsahující jména ulic či názvy měst. Druhá část pak z výstupu první fáze lokalizuje adresné bloky.

2.3.2 Vlastnosti tokenů

K tokenu je ve větě obvykle připojena skupina funkcí, získaná prostřednictvím jednoho či více kritérií z následujícího výčtu:

[12] Java anotační šablonový systém.
[13] Systém extrakce informací na bázi učících se pravidel.
[14] Extrakční systém učící se pravidla z částečně strukturovaných textů.

- Řetězec představující token.
- Pravopisný typ tokenu, který může nabývat hodnot dle různých druhů slov, číslic, speciálních symbolů, mezer, interpunkcí, atd.
- Mluvnický význam tokenu.
- Seznam slovníků, ve kterých se token nachází. Toto kritérium dále může být zdokonaleno uvedením, zda se token nachází na začátku, konci či uprostřed pojmu ve slovníku.
- Poznámky připojené předchozími kroky při zpracování.

2.3.3 Pravidla

Pravidla pro určení jednoduché entity se skládají ze tří druhů vzorů:

- Volitelný vzor zachycující kontext před začátkem entity.
- Vzor odpovídající tokenům uvnitř entity.
- Volitelný vzor pro zachycení kontextu po konci entity.

Příklady:

Vzor pro identifikaci osobního jména z formy *„Dr. Roman Novák"*, tvořené z části titulu, uvedeném v listu titulů (obsahující prvky jako: Prof, Dr, Mr, Ms, ...) , tečky a dvou vlastních jmen:

({ Slovník = Tituly } { tečka } { Pravopis = vlastní jméno }{2}) → Osobní jméno.

Vzor nalezení jména firmy z formy *„The XYZ Corp."*, obsahující nepovinný prvek The (A, An, The), druhým členem jsou vlastní kapitalizované zkratky a třetí, poslední, částí je vlastní jméno, uvedené v listu společností (obsahující prvky jako: Corp., A.s., Comp., ...) :

({ Řetězec = 'The' }? { Pravopis = zkratka } { Pravopis = vlastní jméno, Slovník = znak společnosti }) → jméno firmy.

Pravidla pro označení hranic entity jsou potřebná u některých jejich typů, zejména víceslovných jako například knižní tituly, je účinnější definovat samostatná pravidla pro označení jejího počátku a konce. Ty se samostatně vystřelují a všechny tokeny mezi začátečními a konečnými značkami jsou brány jako entita. Mnoho úspěšných pravidlových extrakčních systémů je založeno na těchto pravidlech, například STALKER[15] či Rapier.

Pravidla pro vícenásobné entity nabývají podobu regulárních výrazů s více sloty, z nichž každý představuje jinou entitu. Tyto pravidla vedou k uznání více entitních subjektů současně a jsou užitečná pro záznamově orientovaná data.

Příklad pravidla, které extrahuje dvě entity, počet ložnic a cenu apartmánu.

({ pravopis = číslo }) : Ložnice ({ řetězec ='BR' }) ({}*) ({ řetězec ='$' }) ({ pravopis = číslo }) : Cena → Počet ložnic = Ložnice, Hodnota = Cena

[15] Učící se extrakční systém analyzující částečně strukturované online texty.

Alternativní formy pravidel: Mnoho *state-of-the-Art*[16] pravidlových systémů umožňuje v obou částech (vzorové i akční) užít libovolné programy, napsané v procedurálních jazycích jako například Java a C++. Například, GATE[17] podporuje programy v jazyce Java v místě svého vlastního pravidlového skriptovacího jazyka, nazývaného JAPE. Tento způsob umožňuje akční části pravidla přístup k různým funkcím, které byly použity ve vzorové části, a jejich užívání ke vkládání nových polí pro anotovaný řetězec.

2.3.4 Organizace sbírky pravidel

Typický pravidlový systém se skládá z rozsáhlé sbírky pravidel a často se jich na stejnou akci, avšak různé druhy vstupu, užívá více. Každá pravidlová střela identifikuje část textu, kterou je tzv. konkrétní entita nebo entitní podtyp. Je běžné, že se části textu, vybrané různými pravidly, překrývají. Tento fakt by bez jakéhokoliv zásahu následně vedl ke konfliktům, proto je velmi důležitou součástí systému řád, jakým jsou pravidla organizována, a posloupnost, ve které jsou použita tak, aby se vzniklé konflikty odstranila. Tento problém tvoří nejvíce nestandardizovanou a potřebám upravitelnou část pravidlového systému, často zahrnující mnohé heuristiky a funkce pro speciální řešení případů.

2.3.5 Pravidla učící se algoritmy

Typický extrakční systém entit závisí na velkém, jemně vyladěném souboru pravidel. Ta jsou dodnes v mnohých z nich často ručně psaná doménovými experty, nicméně v některých případech je možné naučit se pravidla automaticky, a to z označených příkladů entit v nestrukturovaném textu.

Hlavním úkolem je v takovém případě vytvoření a naučení se neseřazených disjunktních pravidel. Systém dostává několik příkladů z nestrukturovaných dokumentů D = x1, …, xN, nazývaných trénovací množinou, kde jsou správně označeny všechny výskyty entit. Soubor naučených pravidel P1 , ..., Pk má vlastnosti takové, že akční část každého pravidla je tvořena jedním z tří typů popsaný v kapitole 2.3.3. Tělo každého pravidla P odpovídá zlomku S(P) dat segmentů v N trénovacích dokumentech, říkáme tomu podíl pokrytí P. Ze všech segmentů P je akce určená pravidlem P správná pouze pro všechny podmnožiny S'(P). Poměr velikostí S'(P) a S(P) je pak přesnost pravidla. V pravidlovém učení je naším cílem pokrýt všechny segmenty, které obsahují anotaci jednoho nebo více pravidel a zajistit, aby přesnost každého pravidla byla vysoká. Soubor pravidel pak musí na vstupních dokumentech poskytovat přesnost a dobré pokrytí. Proto je triviální řešení, které zahrnuje vlastními, velmi specifickými pravidly každý subjekt v trénovací množině D k ničemu, i když má soubor 100% pokrytí a přesnost [2].

Pro zajištění generability se pravidlové učící se algoritmy pokoušejí definovat co nejmenší soubor pravidel, která pokryjí maximální počet trénovacích případů s vysokou přesností. Avšak nalezení takové optimální velikosti souborů pravidel je nemožné, a tak stávající pravidlové učící se algoritmy následují *greedy hill climbing strategii*[18] učení jednoho pravidla po druhém dle následujícího postupu:

[16] Nejvyšší úroveň vývoje.
[17] Java extrakční open source systém.
[18] Matematický algoritmus – http://en.wikipedia.org/wiki/Hill_climbing .

1. RSET = soubor pravidel, zpočátku prázdný.
2. Při existenci entity x ∈ D na kterou se nevztahuje žádné pravidlo z RSET
   a. Vytvoří se nová pravidla kolem x.
   b. Přidají se do množiny RSET.
3. Poprocesní pravidla pro pročištění rerundantních pravidel.

Hlavním úkolem v rámci výše uvedeného postupu je přijít na to, jak vytvořit nové pravidlo, které má vysoké celkové pokrytí, které je nonredundantní k pravidlům, co jsou již v RSET, a které se charakterizuje vysokou přesností. Bylo navrženo několik strategií a heuristik a v zásadě spadají do dvou kategorií: zdola-nahoru nebo shora-dolu. V prvním případě je specifické pravidlo obecné a v shora-dolu strategii je obecné pravidlo specializováno jako propracovaný text.

2.3.6 Shrnutí

Výhodou pravidlových systémů je, že jsou snadno interpretovatelné, lze je rozvíjet a volně rozšiřovat jejich soubor pravidel. Jednou z důležitých složek pravidlových metod je strategie řešení konfliktů. V průběhu let se vyvinulo mnoho různých variant ale jedna z nejpopulárnějších a nejpoužívanějších je strategie uspořádání pravidel dle priority. Většina systémů umožňuje doménovému expertu výběr strategie ze sady několika předdefinovaných. Pravidla jsou obvykle ručně psaná expertem, ale mnohé systémy také podporují automatické učení pravidel z příkladů.

Optimalizace výkonu pravidlového extraktoru: většina pravidlových systémů je založena na regulární gramatice, která může být sestavena jako deterministický konečný automat (DFA) za účelem efektivního zpracování. To znamená, že jeden průchod přes vstupní dokument může být použit k nalezení všech možných pravidlových střel. Každý vstupní token se ve výsledku změní přechodem z jednoho stavu do druhého na základě charakteristik, aplikovaných na vlastnosti připojené k tokenu [3].

Obsah podkapitol 2.2 a 2.3 byl z velké části čerpán ze dvou publikací [2] a [3].

2.4 JAPE

JAPE (anotační šablonový nástroj jazyku Java) je verze CPSL (jazyka specifikujícího společné vzory). Java anotační vzorové jádro, poskytující konečný stav nad anotacemi, tvořený regulárními výrazy.

JAPE umožňuje rozpoznat regulární výrazy v anotacích nad dokumenty. Typicky, jsou regulární výrazy aplikovány na znakové řetězce, jednoduché lineární posloupnosti objektů, zde jsou ovšem užity na mnohem složitější datové struktury. To má za výsledek, že v některých případech je odpovídající proces nedeterministický. (tj. výsledky jsou závislé na náhodných faktorech, jako je adresa, na které jsou ve virtuálním stroji uložena data). Pakliže struktura v odpovídajícím grafu vyžaduje pro rozpoznání více než moc regulárního automatu, JAPE volí alternativu libovolně. V mnoha užitečných případech lze data uložená v anotačních grafech v GATE, či jiných systémech zpracování jazyka, považovat za jednoduché sekvence a spárovat je deterministicky za užití regulárních výrazů [4].

Gramatika JAPE je tvořena několika stádii, z nichž každé se skládá ze souboru vzorových a akčních pravidel. Fáze běží postupně a tvoří kaskádu konečných stavů nad anotacemi. Levé strany (LHS) pravidel se skládají z popisu anotačních vzorů a strany pravé (RHS) se obsahují anotační manipulační příkazy. Anotace vyhledaná na LHS pravidla může být přiřazena RHS pomocí štítků, které jsou připojeny ke vzorovým prvkům [4].

Příklad pravidla pro určení zaměstnání:

```
Phase: Jobtitle
Input: Lookup
Options: control = appelt debug = true
Rule: Jobtitle1
(
  {Lookup.majorType == jobtitle}
  (
    {Lookup.majorType == jobtitle}
  )?
)
:jobtitle
-->
:jobtitle.JobTitle = {rule = "JobTitle1"}
```

Obrázek 2.2: Pravidlo anotačního jazyka JAPE

2.5 Reprezentace znalostí

V předešlé části kapitoly jsme se zabývali otázkou metod získávání strukturovaných informací. Na konci procesu uspořádání a analýzy se informace mění ve znalost, což je v definici reprezentace vzájemných vztahů entit a operací, které je s nimi možné provádět. Jedná se o množinu nějakých konvencí, popisujících třídu objektů. Reprodukce takových informací je ve své podstatě způsob vkládání explicitní informace do báze znalostí, za užití jazyků pro reprezentaci nebo tzv. schémat reprezentace.

Základní typy reprezentace jsou dva:

- Procedurální – týkají se postupů a způsobů, doporučení a předpisů jak dosáhnout plánovaného cíle.
- Deklarativní – znalosti deskriptivní (popisné) či konstatující.

V pravidlových systémech pak rozlišujeme tři části, báze dat (obsahující fakta), báze pravidel (znalostí) a inferenční mechanizmus neboli interpret, poskytující návod na výběr pravidel z konfliktní množiny. Na výstupu daného procesu je určitý výsledek, který je nutné nějakým způsobem reprodukovat. Hlavním jazykem, sloužícím pro reprezentaci znalostí, je RDF (Resource Definition Framework). Jedná se především o jednoduchý databázově orientovaný jazyk, v němž je možné vyjádřit tvrzení typu *„Objekt X nabývá pro vlastnost Y hodnoty Z“* za užití takzvaných *trojic* (“triple”) *subjekt – predikát – objekt.* Například *„Vaclav Havel is a name“* či *„Lockheed SR-71 Blackbird is an aircraft“*. Díky jazyku RDF Schema, který se specializuje na tvorbu jednoduchých klasifikačních schémat, je tento systém vhodný pro reprezentace slovníků [5].

3 Návrh

3.1 Základní problémy

V úvodu druhé kapitoly byly uvedeny důvody potřeb na přeměnu neformátovaného textu na formátovaný a následné extrakce informací z něj. Nyní máme za úkol vytvořit jednoduchý systém, analyzující text a přeměňující vstupní nestrukturovaná data na výstupní strukturované znalosti.

Zadání práce nám přesně nedefinuje variantu samotné extrakce, avšak pro jednodušší a efektivnější implementaci považujeme za příhodné užití pravidlového systému. Ten, jak již bylo v dosavadním textu popsáno, má ručně definovaná či učená pravidla, která se ve většině případů popisují pomocí regulárních výrazů. Předpisy v našem systému z nich nebudou pouze čistě vytvořena, avšak právě výrazy budou jejich nedílnou a důležitou součástí. Pro přesnější identifikaci entit a vztahů budou v některých fázích průchodu aplikace užity pomocné datové struktury, zejména slovníky. Pro rychlé a efektivní řešení právě těchto dvou hlavních problematik jsou vhodné skupiny skriptovacích jazyků. V našem případě se jedná o jazyk Python, jehož doménou jsou především datové struktury a díky knihovně *re* také problematika regulárního jazyka. Aplikace musí klást důraz na další kritéria jako je jednoduchost, přesnost, rychlost či pokrytí. Tyto dvě poslední veličiny spolu nepřímo souvisí, a jelikož očekáváme vstupní data o velikosti gigabytů, musí program být schopen takové množství dat rychle analyzovat. Této vlastnosti je však velice obtížné dosáhnout bez ztrát na informační úplnosti. Více rozvedeno v kapitole 3.4.

Vstupní data budou tvořit články z informačního serveru Wikipedie. Jelikož je tento systém otevřený, nalézají se na internetu volně ke stažení jeho kompletní databáze. Jednotlivé informace o tématech jsou do souboru vkládány ve formátu XML, samotné texty článků však mají svou vlastní originální syntaxi značkovacího jazyka wiki. Abychom se dostali k čistému obsahu článků, budeme muset tento problém řešit.

3.2 Použité nástroje

Takto naformátovaná databáze by byla náročná na analýzu, a tak z ní bude potřeba vyjmout a upravit jen ty nejdůležitější části jako jsou název článku a jeho obsah. K tomuto kroku máme možnost využít služeb skriptu ***wikiparser.v4***, vytvořeného autory Davidem Smejkalem a Ing. Markem Schmidtem. Ten z wiki databáze extrahuje čistý text bez metadat a informací, zdali se jedná o nadpis, podkapitolu či samotný obsah článku, proto jej budeme nuceni v několika částech poupravit tak, abychom od sebe mohli jednotlivé položky a části textu oddělovat. V další fázi bude zapotřebí provést lexikální analýzu jednotlivých lexémů textu, pro co nejpřesnější určení slovního významu tokenů je užit volně dostupný anotační program ***Tree-Tagger***, který vstupní text analyzuje a pomocí vlastních knihoven a databází určuje slovní druh a lemmu. Výstupem tohoto nástroje již bude soubor, zpracovávaný vytvořeným skriptem.

3.3 Návrh a definice jazyka

Program Tree-Tagger určuje a třídí lexikální významy slov (anglicky part of speech, POS) do více jak třiceti druhů, z nichž pro každý z nich musíme definovat vlastní název. Více o aplikaci a problematice POS v kapitole 4.3.2. Jednotlivá slova námi navrženého a užívaného jazyka byla vygenerována jako zkratky anglických odborných výrazů pro slovní druhy a značky, vytvořené již zmiňovaným programem. Například ‘proper noun-singular’ (podstatné jméno vlastní) má v našem jazyce značku NP, plurální forma potom NPP, ‘determinant’ – DT, ‘adverb’ – ADV a tak dále.

V následující tabulce uvádíme výčet těch nejzákladnějších lexikálních druhů slov :

Lexikální význam	Příklad	Značka	Lexikální význam	Příklad	Značka
Podstatná jména (singular)	church, apple	NN	Číselné řetězce	0, 3.14, 365	NUM
Postatná jména (plural)	cars, houses	NNP	Vlastní Zájmena	I, me, you	PNPE
Podstatná jména vlastní (singular)	Washington, Peter	NP	Příslovce	enough, nicely	ADV
Podstatná jména vlastní (plural)	Rails, Netherlands	NPP	Determinanty	the, this, that	DT
Přídavná jména	happy, lucky	ADJ	Spojky	in, of	IN
Slovesa	be, walk, run	VRBB	Koordinační spojky	and, but, or	CC
Slovesa v třetí osobě	is, has	VRB3D	Znaménka ukončující větu	. ? !	DOT
Slovesa v minulém čase	was, were	WRBD	čárka	,	COMMA

Tabulka 3.1: Vnitřní jazyk aplikace

Ve většině funkcí aplikace budou analyzovány věty jako celek, a tak v nich budou generovány a analyzovány řetězce, skládající se z prvků našeho jazyka. Více o datových strukturách v kapitole 4.3.3.

3.4 Rychlost vs. pokrytí

Velikost anglické databáze Wikipedie, obsahující v danou chvíli přeš 3.700 000 článků, činí v původní XML formě více jak 28GB. V upravené, vyparserované verzi se sice toto číslo citelně zmenší, a to na necelou polovinu, avšak i rozbor takového množství dat je časově velice náročný. Protože jedním z kritérií aplikace je důraz na rychlost, budeme nuceni tento problém řešit. Při úvaze nad fungováním vyhledávacího jádra Wikipedie jsem dospěl k závěru, že při neúspěšném hledání na internetovém portálu server zobrazí pouze články, ve kterých je hledaná informace obsažena v titulku, nikoliv seznam témat, kde se hledaný token objevuje uvnitř textu. Už z principu a hlavního úkolu wiki informačních systémů se tento způsob zdá jako správný, navíc rychlý a přehledný. Chceme-li hledat základní informace o určité tématice, je pro nás daleko přínosnější existence článku, specializovaného na tuto otázku, než výčet stránek, ve kterých se hledaný řetězec vyskytuje pouze jako součást věty. Proto jsme se rozhodli jít touto cestou a pro efektivnější hledání vytvoříme

pomocný soubor, obsahující seznam názvů článků, extrahovaných z databáze, k nim bude následně přiložena číselná informace o pozici, na které se text nachází. Tato funkce dovolí skriptovacímu jazyku Python rychle nalézt daný text, následně jej vyjmout a analyzovat. Nevýhodnou se tato varianta stane, pokud hledanému termínu neodpovídá žádný článek a tím pádem o něm nezískáme žádné znalosti. V opačném případě je zde vysoké procento pravděpodobnosti, že nejdůležitější popisné informace budou úspěšně nalezeny.

3.5 Konečný automat

Protože počet stanovených pravidel v extrakčních systémech je vždy konečný, je možné popsat daný systém pomocí konečného automatu, definovaného regulárními výrazy. Pro samotnou implementaci však tato metoda zvolena nebyla a pravidla našeho systému jsou psána regulárním jazykem pouze z části.

Hlavním důvodem tohoto postupu je fakt, že Python nám tímto způsobem umožňuje jejich pohodlnější vývoj a samotný programovací jazyk je lépe využit. Pro funkci extrakce tak dostáváme lepší Turingův stroj[19].

[19] Teoretický model počítače.

4 Implementace programu

4.1 Prostředí

Skript byl vytvářen a testován v prostředí linuxové distribuce Ubuntu[20]. Hlavním důvodem užívání tohoto operačního systému byly již předešlé zkušenosti s ním a uživatelsky přívětivé prostředí pro vývoj skriptových aplikací. Jazyk Python je užíván ve verzi 2.6 a to z důvodu kompatibility s knihovnami, školním serverem merlin a přiloženým skriptem *wikiparser*.

4.2 Chování aplikace

Program funguje na principu konzolové aplikace a námi hledaný libovolný vstupní token se zadává jako argument v příkazovém řádku. Extraktor vložený vstup analyzuje a po úspěšném vyhledání článku a jeho rozboru vytiskne o daném tématu na výstup nalezené základní informace. Text finálních znalostí je ve formátu RDF, skládající se ze tří částí. Dva prvky jsou tvořeny popisy entit a tím třetím je vztah mezi nimi.

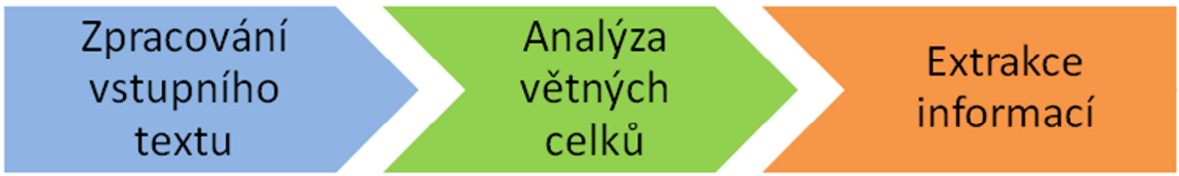
Příklad výstupu programu pro hledaný řetězec *Cascada*:

```
./wikiextractor.py –t Cascada
Cascada  consist of     German born English singer Natalie Horler
Cascada  consist of     Record producer DJ Manian
Cascada  consist of     Yanou
Cascada  is       German eurodance group
Cascada  is a     group
```

Obrázek 4.1: Výstup aplikace

4.3 Části skriptu

Tato podkapitola je věnována jednotlivým důležitým funkcím skriptu, které jsou podrobně vysvětleny a pro názornost připojeny také příklady užití.



Obrázek 4.2: Základní procesy aplikace

[20] Linuxová distribuce operačního systému.

Problematika zpracování a rozboru vstupního textu je rozebrána v sekcích 4.3.2 až 4.3.3, analýzou, úpravou a selekcí větných celků se zabývají podkapitoly 4.3.5 a 4.3.6 a finální extrakce základních znalostí z textu za užití pravidel je popsána v článku 4.3.8.

4.3.1 Ovládání aplikace

Chod skriptu je řízen pomocí příkazů, zadaných při spuštění jako jeho parametry. Název hledaného článku se vkládá po příkazu –t, a protože se mnohdy jedná o celá sousloví, můžeme i my zadávat víceslovná spojení. U těchto složitějších případů je velice těžké, trefit se do přesného názvu článku, například *„People's Republic of China“*. Vyhledávače Wikipedie či Googlu užívají složité vyhledávací algoritmy a přesměrování, my se musíme spokojit s výčtem podobných témat, ze kterých si uživatel může správný název dohledat, tato funkce je aktivovatelná příkazem –a. Výčet parametrů je znázorněn v následující tabulce :

Přepínač	Argument	Popis
-t	Token	Řetězec (i víceslovný), jehož popis extrahujeme
-q		*Rychlé hledání – pouze přesný název článku
-c		*Pomalé hledání – vypisujeme seznam názvů podobných článků
-a		**Celé věty, obsahující informace o tokenu se vypíší na výstup
-o		**Výsledné znalosti se vypisují do souboru ‘OUTPUT.TXT’
--h		**Výpis nápovědy

Tabulka 4.1: Parametry aplikace, * : jeden z prvků povinný, ** : nepovinné parametry

Příklady spuštění aplikace:

```
#./wikiextractor.py –q –o Anarchism
#./wikiextractor.py –q –a –t United Kingdom
#./wikiextractor.py –c –t The Tower of London
```

4.3.2 Předzpracování dat

Vyhledaný text je vyjmut z databáze, protože jeho přímé vkládání na vstup programu Tree-Tagger v některých případech z důvodu obsahu nestandardních znaků skončilo chybou, ukládáme prvně úvodní odstavec nalezeného článku do záložního souboru `temp/dbtext.txt`. Až samotný obsah tohoto objektu je podroben analýze slovních druhů anotačním programem. Výstup značkovací aplikace je přesměrován a uložen do druhého pomocného souboru `temp/tttext.txt`, jeho obsah je následně postupně načten extrakčním skriptem do paměti a tím naše práce s externími soubory končí.

Formát textu má po analýze pro jednu slovní jednotku podobu [slovo]\t[POS]\t[lemma]\n. Anglická zkratka *POS* (part-of-speech) zde znamená v podstatě lexikální slovní druh. *Lemma* pak určuje základní podobu (či slovníkový tvar) lexému neboli slova či fráze. Pro názornost demonstrujeme reakci programu Tree-Tagger na větu *„Hello, nice to meet you.“*:

slovo	POS	lemma
Hello	UH	Hello
,	,	,
nice	JJ	nice
to	TO	to
meet	VB	meet
you	PP	you
.	SENT	.

Tabulka 4.2: Názorný výstup aplikace Tree-Tagger

4.3.3 Datové struktury

Z takto uspořádaných dat potřebujeme získat nejen samotná slova, ale i k nim přiřazené lexikální informace, proto řádek po řádku načtený text procházíme a pomocí regulárního výrazu v podobě `(.*)\t(.*)\t(.*)\n` extrahujeme všechny tři obsažené tokeny. Následně podle druhé skupiny, obsahující POS informaci, třídíme jednotlivá slova dle významu a vkládáme je do předem připravených globálních polí. Ta užíváme v celém programu dvě – pole WORDS, do kterého řadíme samotné lexémy, a pole CLASSES, obsahující slova námi definovaného jazyka, vkládané podle POS informace. Před rozšiřováním první z těchto dvou zmiňovaných datových struktur o další prvek však nejdříve určujeme, o jaký slovní druh se jedná. V případě podstatného jména či determinantu do pole vkládáme místo lexému jeho lemmu (obsah třetího sloupce). Tento krok děláme z důvodu unifikovanosti a snazší, efektivnější extrakce. Za zmínku stojí také lokální řetězec, předávaný mezi funkcemi, obsahující záznamy o lexémech, vytvořené v námi definovaném jazyce. Tuto strukturu jsme zavedli kvůli regulárním výrazům, které užíváme právě na tyto data. Obsahu řetězců je ve speciálním formátu [číslo]:[literál]\s, kde číslo udává pozici slova ve větě a literál, jak je již známo, značí POS druh a dvojtečka plní funkci oddělovače mezi těmito hlavními jednotkami. Takováto struktura je vytvořena především kvůli snadnějšímu odkazování a prohledávání, hlavní užití bude popsáno v následujícím odstavci.

Názorná ukázka obou polí po vložení věty *„Information extraction from unstructured texts is a difficult task.“*

Pole WORDS : ['information', 'extraction', 'from', 'unstructured', 'text', 'is', 'a', 'difficult', 'task', '.']
Pole CLASSES : ['NN', 'NN', 'IN', 'ADJ', 'NNP', 'VRB3RD', 'DT', 'ADJ', 'NN', 'DOT']
Řetězce pro regulární výrazy : 0:NN 1:NN 2:IN 3:ADJ 4:NNP 5:VRB3RD 6:DT 7:ADJ 8:NN 9:DOT

4.3.4 Počáteční analýza objektů

Po načtení a rozboru struktury textu jej musíme připravit na podrobnou analýzu stavby a rozložení do jednotlivých větných bloků, procesy, které jsou podrobně popsány v následující části 4.3.5. Abychom mohli data poslat dál, je nutné v nich určit jednotlivé prvky. I bez aktuální znalosti následujícího textu uvedu příklad – při větě obsahující termín *„United States of America“* bude dále pro náš skript důležité, aby tento výraz nezůstal jako posloupnost podstatných jmen vlastních

oddělených předložkou, nýbrž aby byl označkován celý jako objekt. Dále, tento funkční blok přispívá ke snadnějšímu určení větného podmětu a přísudku, ale o těchto problematikách až v následující podkapitole.

Pro definici objektů v této funkci užíváme, po studii a rozborech anglických textů, námi napsaná pravidla (regulární výrazy) a také slovníky. Je vybráno několik hlavních témat pro určení třídy objektu a pro každé z nich je vytvořena vlastní datová struktura, naplněná pomocnými daty. Při zkoumání většiny oborů využíváme existence právě těchto seznamů, vytvořených na začátku skriptu. Pakliže se posloupnost větných členů shoduje s regulárním výrazem, obsaženým v jednom z pravidel, je daný text hledán v příslušném slovníku. Pro příklad uvedeme část věty *„in the Czech Republic“*, při průchodu výrazy dojde v jednom z nich, označujícím státy, ke shodě. Nalezený výraz *„Czech Republic“* je poté předložen slovníku *COUNTRIES*, obsahujícím seznam všech světových států a v případě shody je daný token označen jako *valid_state*. V některých situacích je místo pomocných datových struktur užit pouze sekundární regulární výraz, jako v případě určení data, kde se při posloupnosti numerických či alfanumerických celků, oddělených interpunkčními znaménky na závěr ověří validita shodou s výrazem *ReDate*. Takto dále, krom dvou již zmiňovaných oborů, zkoumáme text na jména (+ příjmení), čas, url, množství (amount of) a zbylé nerozpoznané objekty. I když jsme při původním návrhu této části programu přisuzovali větší míru důležitosti, než v závěrečné podobě skriptu opravdu má (především v počtu zkoumaných témat a vytvořených slovníků), jsou ty hlavní objekty skutečně potřebné.

K samotné struktuře a formátu zkoumaného textu, uvedených v předcházejícím odstavci, při užití regulárních výrazů, nám funkce *re.findall* vrací pole všech nalezených validních posloupností a při obsahu více termínů, majících stejnou strukturu, na ně následně můžeme obtížně zpětně odkazovat. Proto je před každý člen přidána i numerická hodnota pozice prvku ve větě, abychom mohli jednoduše na odpovídající části řetězce odkazovat, nebo jej případně modifikovat.

Pro názornost uvedu ukázku na textu *„Moscow is the capital.“*.
Tato věta se v řetězci zobrazí jako ‘0:NP 1:VRB3RD 2:DT 3:NN 4:DOT‘. Po průchodu funkcí nám regulární výraz pro objekt vypíše shodu na prvcích [('0', 'NP '), ('2', 'DT 3:NN ')], kde díky číslům, označujícím počátek nalezených prvků, můžeme globální pole lehce upravovat. V tomto případě by bylo možné řešit situaci lehčeji, avšak při nálezu dvou stejně složených posloupností se stává tento způsob efektivním.

4.3.5 Problematika zpracování a stavby vět

V každém jazyce, angličtinu nevyjímaje, se v textech vyskytují souvětí, obsahující věty vsunuté, věty, ve kterých je podmět nevyjádřený či uveden v nepřímé řeči a velkou škálu dalších odlišností. Čím bohatší je jazyk, tím mohou být souvětí košatější a jejich stavby a slovosledy složitější, což znamená pro vědu zabývající se extrakcí informací z přirozeného jazyka komplikace. Ať už se jedná o jazyk český či anglický, druhy vět, jako je podmětná, přísudková, předmětná a množství dalších, se vyskytují všude a pro správné zapojení a chápání v kontextu má každý druh své specifické vlastnosti. Přestože je detekce a analýza této problematiky implementačně a vývojově velice složitá, po průzkumu a studiu jsem vyvodil určitá základní pravidla pro řešení aspoň v těch nejběžnějších a nejjednodušších případech.

V první řadě však musíme text rozdělit na větné celky, věty a souvětí ukončené tečkou by byly pro extrakci informací příliš slabé, a proto musíme jít hlouběji. Proto byly přidány do slovníku

větně-hraničních výrazů slova jako *comma*, *or* a *but*, abychom text rozdělili na co nejjemnější větné části. I toto řešení není ideální a přináší komplikace a složitosti, o kterých budou následující odstavce.

Věty, ve kterých je podmět i přísudek vyjádřen (dále je toto spojení popisováno jako báze), jsou brány jako hlavní a tyto dva základní prvky jsou uloženy do paměti pro jejich následnou případnou potřebu. Pro rychlejší vyhledávání je nalezený podmět zkoumán, obsahuje-li hledaný výraz, abychom v případě neúspěchu zabránili zbytečné další analýze a ušetřili tak čas. V pozitivní verzi je věta dále analyzována, a to její složitost, tento faktor rozdělujeme do tří tříd a určujeme je podle počtu větných úseků oddělených slovy *of*, *in*, *by*, *to*, *at* či *for*. Prvním ze tří stupňů jsou označeny věty, obsahující bázi a k ní pouze slovní spojení neobsahující žádné z hraničních výrazů, příklad „The apple is a pomaceous fruit.“. Věta symbolizující druhou třídu má oproti té první v sobě jednou obsažen oddělující výraz, příkladem věta „A flower is the reproductive structure found *in* flowering plants“. Poslední, třetí, verze se pak skládá ze dvou těchto hraničních typů : „Television is a widely used telecommunication medium for transmitting and receiving moving images“. Tyto informace jsou rovněž zaznamenány a jejich další možné užití bude popsáno v následujících odstavcích.

Na opačném konci stojí věty, ve kterých tyto dva základní bázové prvky obsaženy nejsou, nebo chybí-li aspoň podmět. V tuto chvíli ovšem nevíme, jedná-li se o větu rozvíjející, neúplnou, s nevyjádřeným podmětem a přísudkem, či pouze o několikanásobný větný člen oddělený z obou stran větně-hraničními výrazy. Řešením detekce několikanásobného větného členu a věty rozvíjející, i když ne vždy správným, je shoda s regulárním výrazem je popisujícím, shoda slovních druhů výrazů před a po hraničním prvku a informace, nachází-li se text uvnitř souvětí. Obě tyto problematiky jsou pak dále řešeny stejným způsobem. V prvním bodě je analyzována, stejně jako v případě věty hlavní, složitost textu, jediným rozdílem je nepřítomnost báze, tudíž je tato vlastnost zkoumána pouze do druhé úrovně. V následující fázi se v přepínači srovnává složitost hlavní věty a aktuálního textu, výsledkem této funkce je informace, jak velkou část textu budeme nastavovat k hlavní větě. Následně se ono samotné spojení provede a na závěr se opět analyzuje složitost výsledného textu a dané řetězce se uloží do paměti. Pro lepší pochopení tohoto bloku uvedeme dva příklady:

Souvětí : *Monte Blanc is the highest mountain in the Alps*{věta hlavní : složitost 2} *and*(hraniční symbol) *2nd highest mountain in Europe*{věta vedlejší : složitost 2}.
Výsledek : *Monte Blanc is the highest mountain in the Alps. Monte Blanc is 2nd highest mountain in Europe.*

Souvětí : *London is the capital of England*{věta hlavní : složitost 2} *and*(hraniční symbol) *the United Kingdom*{věta vedlejší : složitost 1}.
Výsledek : *London is the capital of England. London is the capital of the United Kingdom.*

Druhou problematiku, situaci kde věta postrádá předmět či celou bázi, řešíme přidáním chybějících článků z báze, vytvořené z předchozího textu, na začátek aktuální věty. Takto ovšem může nastat několik sporných stavů, například obsahuje-li text sloveso pouze v minulém čase prostém či gerundu (koncovky –ing a -ed). V takovém případě musíme zkontrolovat uložený bázový predikát, obsahuje-li sloveso ‘be‘ a v kladném případě jej spojíme se slovesem věty vedlejší.

Souvětí : *The Arctic Ocean is located in the Northern Hemisphere*{věta hlavní} *and*(hraniční symbol) *mostly in the Arctic north polar region*{věta vedlejší}.
Výsledek : *The Arctic Ocean is located in the Northern Hemisphere. The Arctic Ocean is mostly in the Arctic north polar region.*

V poslední fázi znovu podrobíme aktualizovanou větu rozboru složitosti a výsledné informace uložíme.

4.3.6 Problematika zájmen a nepřímé řeči

Další komplikací, vyskytující se téměř ve všech článcích, která může mít zásadní dopad na výsledek extrakce, je nepřímá řeč. V takové větě je hlavní objekt (předmět) zastoupen zájmenem. V regulérní hlavní větě, obsahující bázi, hledáme pro její určení vztah podstatné jméno + sloveso, v případě nepřímé řeči je to pak posloupnost zájmena a slovesa. Při čemž máme definovaný slovník, obsahující nejběžnější validní zájmena, a díky kterým určíme číslo. Pokud se daný výraz objeví, přepíšeme nalezené zájmeno, podmětem, uloženým v bázi z dřívějších vět.

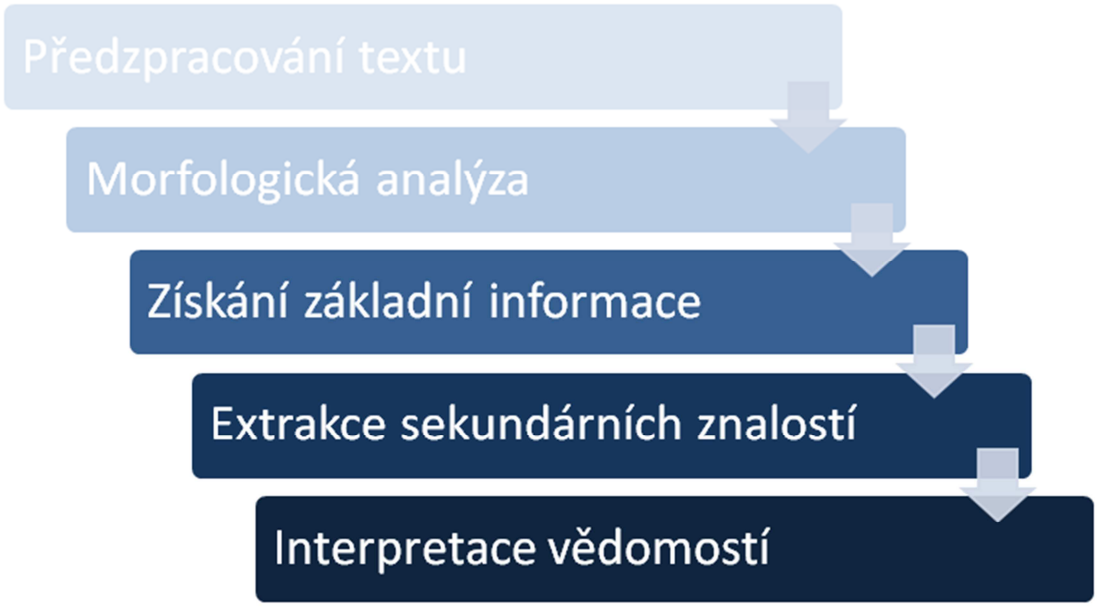
Souvětí : *The Chernobyl disaster was a nuclear accident*{věta hlavní} *that*(hraniční symbol) *occurred on 26 April 1986 at the Chernobyl Nuclear Power Plant in the Ukrainian*{věta vedlejší}.
Výsledek : *The Chernobyl disaster was a nuclear accident. The Chernobyl disaster occurred on 26 April 1986 at the Chernobyl Nuclear Power Plant in the Ukrainian.*

4.3.7 Částečná kontrola validity věty

Do této fáze jsme kromě kontroly při analýze složitosti vět zatím nijak nezkoumali jejich správnost slovosledu a členů a především při třetím způsobu modifikace textu může nastat, že se k větě vedlejší přidá báze na místo, kde by poté výsledek nemusel dávat správný význam. Může se také stát, že vstupní text obsahuje zvláštní větné formy a seskupení, se kterými jsme se při vývoji skriptu nesetkali, a které by mohly přes funkci větného editoru přejít v neúplné podobě, například jako pouhý podmět s přísudkem. Protože chceme zabránit, aby se na výstupu objevoval takovýto nesrozumitelný a žádnou informaci nesdělující text, je vytvořeno jakési síto, v podobě velkého regulárního výrazu, obsahujícího kombinace velkého množství základních větných struktur. Tato funkce byla původně vyvíjena pro širší využití, a tak funguje i jako nůž, který větu rozděluje podle hraničních výrazů, popsaných v třetím odstavci podkapitoly 4.3.5 na několik částí, z nichž na závěr utvoří výslednou větu v textové podobě. Pokud má text neznámou formu a skládá se z posloupnosti slovních druhů, které neodpovídají výrazu, je zahozen. I přes neustálé úpravy a rozšiřování primárního regulárního výrazu není tato technika stoprocentně úspěšná.

4.3.8 Hlavní extrakční analyzátor entit

Předešlé podkapitoly se zabývaly otázkami předzpracování textu za pomoci externích aplikací a morfologické analýzy jednotlivých větných prvků, upravující data do zpracovatelné podoby. Nyní je na řadě samotná otázka extrakce informací z nestrukturovaného textu. Hlavní tři body této problematiky jsou zobrazeny v tmavě zbarvených bublinách na obrázku a v následující části budou popsány a vysvětleny.



Obrázek 4.3: Jednotlivé kroky extrakce

Počáteční rozbory textu analyzovaly jako vstupní data slovní druhy (přesněji POS informace) a užívaly pro své potřeby mnohdy rozsáhlé slovníky. Tento způsob byl velice efektní, avšak při širším okruhu témat a potřebě podrobnějšího popisu entit může být i časově náročný. My se proto v hlavní části aplikace zaobíráme extrakcí informací přímo z čistého textu a bez užití pomocných datových struktur, čili bez jakékoliv vědomosti, kterou bychom znali před počátkem analýzy. Takové řešení je implementačně daleko zajímavější, ale na druhou stranu také obtížnější a na první pohled nepřesnější. To vše převážně z důvodu, že server Wikipedie je zcela otevřený veřejnosti a i když pro tvorbu článků využívá svou vlastní syntaxi, nikoliv tak pro samotnou strukturu textu. Neužívá také žádné scanovací a kontrolující algoritmy obsahu, a tak se do databáze ukládají texty v takové podobě, jaké je autor vytvoří. Je tedy obtížné navrhnout přesná pravidla, která by měla vysokou účinnost a pokryla veškeré odchylky, které mohou vinou různých stylů jazyka vzniknout.

Pro porozumění textu a správnému rozložení vět, má funkce *mainScanner* předdefinované regulární výrazy, popisující jednotlivé druhy slov jako jsou podstatná jména obecná, vlastní, číslovky, a zkratky, spadající pod skupinu *Object*, dále pak výčet nejužívanějších předložek a spojek, patřící do třídy *Separator*, a na závěr výraz hledající všechny tvary a časy slovesa *„be“* a slovesa v třetí osobě, končící na *–s*, tato skupina spadá do kategorie *Predicate*. Z těchto složek je vytvořen velký, primární regulární výraz, tvarující základní podobu věty a v případě shody rozdělující jí na několik částí, které již slouží k samotné analýze entit a vztahů mezi nimi.

Výčet posloupnosti částí: Subject – Predicate – Object (– Separator – Object)*

Dělení textu probíhá naznačeným způsobem, při kterém se odtrhnou řetězce jmenných slov od slovesného tvaru, obsaženého ve skupině Predicate a od předložek a spojek spadajících pod Separator. Tímto krokem oddělujeme samotné entity od termínů popisujících vztahy a rozšiřujících bázi znalostí o nich. Uvedeme názorný příklad dělení:

Věta *„Pine is tree in the the family Pinaceae“* se po provedení operace rozporcuje na části:
'Pine', 'is', 'tree', 'in', 'the family Pinaceae'.

S takto upravenými informacemi již můžeme provádět analýzu jednotlivých extrahovaných entit a prvků popisujících vazby mezi nimi.

Již od separace a úpravy vět ze souvětí kontrolujeme, aby byl hledaný výraz součástí předmětu, tím pádem jej budeme mít vždy obsažený v části první. Dále následuje slovesný přísudek, ten může nabýt podobu nejen tvarů popsaných ve skupině Predicate, avšak pokud nenásleduje jako další člen předložka či spojka, je možné vytvořit jej i složený, končící na *–ed* (has been called, was amazed, …).

Musíme zdůraznit informaci, že systém analyzuje pouze věty, obsahující v predikátu tvar slovesa „*be*“. Cílem aplikace je nashromáždit o dané problematice základní informace ve tvaru *entity is entity*. Je možné zkoumat všechny věty bez váhy tvaru přísudku, avšak texty, ve kterých je tvořen jinými slovesy ve většině případů přináší pouze informace vedlejší.

Nyní již k pravidlům, jako první systém nahlíží, zdali se hledaný výraz již nenachází ve slovníku, pokud ne, zkoumá se samotný subject, neobsahuje-li již on sám o sobě důležitou informaci, jako například termín *the Golden Bridge*, který již sám nese informaci o tom, že se jedná o most.

V případě neúspěchu je výraz postupně předhazován sérii základních předzpracovávajících pravidel, která každé zastupuje určitou tematickou skupinu a které obsahuje výčet hlavních slovních zástupců, například pravidlo plant obsahuje názvy *tree*, *flower*, *herb* a *bush*. Tento postup využívá nepsaného pravidla, že základní a nejvýše řazený poznatek o tématu je uveden v prvních větách a v základním prostém tvaru. Pakliže se v jednom z předpisů nalezne shoda, je jednak o hledaném tokenu známá první informace Token is Y, ale pro každé téma máme vytvořeny speciální pravidla, a tak víme, na jaká z nich se máme v následujícím textu zaměřit. Systém zaměřený na specializaci pravidel pro každou skupinu témat zvlášť jsme zavedli proto, že užívání všeobecných předpisů by mohlo vést k velkému množství špatně extrahovaných informací. Nenalezli bychom shodu s žádným z primárních pravidel, vzalo by se jako validní poslední slovo z druhé entity, z věty *„the Golden Bridge is an old bridge in London“* by to byl výraz bridge. Vně většině případů se takto povede vytvořit správnou asociaci, pouze při obsahu víceslovného, jmenného subjektu dochází ke ztrátě informace.

Po prvním bodě, hledajícím základní popis entity, následuje druhá, rozsáhlejší část, obsahující sady pravidel pro extrakci základních poznatků. Každé z nich má určitý význam a specifikuje svou vlastní informaci. Jednotlivé předpisy jsou definované a složené převážně z několika jednoduchých regulárních výrazů, z nichž každý se dotazuje jinou část věty.

Příklad jednoho z pravidel určujícího polohu objektu :

```
if TOPIC == 'building':
    if re.search(reBuilding, L1) and re.match('in|of', c1) and re.findall
('(?:(?:(?:a|the)\s)?'+ capital +')', l2):
        MAINDICT[TOKEN + ' is situated in ' + L2] = TOPIC
```

Obrázek 4.4: Příklad extrakčního vzoru

Výsledná metadata o objektu jsou : The Golden Bridge is situated in London.

Následně je daná znalost uložena do slovníku a tam čeká do konce procesu extrakce. Tuto datovou strukturu jsme zvolili jednak kvůli rychlejším operacím oproti poli, ale především z důvodu duplicit, získáme-li informaci, která již je ve slovníku obsažena, je tato jeho část pouze aktualizována. Pro kontrolu duplicit v poli bychom museli vytvářet složité algoritmy, které by snižovaly rychlost

aplikace. Na závěr se děj přesunuje do funkce *getResults*, kde je obsah slovníku analyzován, vložen z důvodu seřazení informací dle abecedy do pole (v důsledku se řadí až data popisující vztah mezi entitami) a dle požadavků vložen na výstup.

5 Testování pravidel

5.1 Vstupní data

Pro experimentování a získání informací o efektivnosti programu byla jako vstupní data užita databáze anglické Wikipedie (přesněji *enwiki-20100916-pages-articles.xml*), aktuální ke dni 16. 9. 2010, upravená skriptem *wikiparser.v4*. Nejspíše pro extrémní velikost souboru v původním XML formátu se nám jej ani po několika pokusech nepodařilo naformátovat celý, a tak jsme byli nuceni užít pouze část databáze s necelou polovinou všech obsažených článků (cca 1,78 milionu).

5.2 Zkoumané oblasti

Jelikož je záběr témat, obsažených v databázi vstupního souboru, rozsáhlý, a vytváření a validace pravidel, pro každé či jen ty nejobecnější a nejběžnější z nich, by byl prakticky nekonečný proces, byly pro implementaci vybrány z celé té škály okruhů pouze dvě oblasti zájmu, a na ty jsem se při experimentování se systémem zaměřil. Tento přístup nám umožňuje testovat program nad velkým množstvím dat, majícím společný základní význam. Jinými slovy takovýmto způsobem podrobíme odpovídající pravidla širokému vzorku textů, a tak dostaneme hodnotnější informace o jejich funkčnosti a účinnosti. Zaměřením se na široký rozsah pravidel, by bylo při pokusech každé využito pouze několikrát, avšak při pokusech nad vybranou tématikou je můžeme aktivovat a analyzovat častěji a získat tak o nich věrohodnější informace, i když bude vzorek pravidel úzký.

Po zvážení a analýze již napsaných předpisů jsem vybral jako oblasti zájmu témata experimentů z geografické oblasti, a to *města* a *státy*. Seznam užitých pravidel bude vypsán v tabulkách u každé třídy zvlášť. Jako výsledné faktory pak budeme analyzovat jejich přesnost a v některých případech pokrytí. Výsledné informace o těchto dvou veličinách budou vycházet z výsledků a následných rozborů experimentů a budou vyjádřeny v procentuálním formátu.

5.2.1 Extrakce informací o městech

Každá aglomerace je známá a výjimečná díky jiným vlastnostem, a tak jsme se snažili najít takové informace, které by měly články v této tematické problematice společné. Mezi hlavní a nejvíce užitečné údaje patří bezesporu situování místa a počet jeho obyvatel. Na tyto základní poznatky jsme se při analýze a výstavbě pravidlových větví zaměřili. Jelikož podle těchto faktorů rozeznáváme celou škálu měst, tak bychom spíše než tento pojem měli užívat označení obydlená oblast, zahrnující pod svá křídla nejen velké celky, ale i vesnice. V otázce situování pak rozdělujeme pojmy na politické (stát, provincie, ...) a geografické (ostrov, řeka, ...).

Pravidlo X is a city (regCity = re.compile('\s?(city|capital|town|village)\s?', re.I))
if re.search(regCity, TOKEN):
Příklad : New York *city* is situated on the East Coast.
elif re.findall(regCity, string):
Příklad : Ottawa is the *capital city* of Canada.
Pravidlo X lies in Y (regCity = re.compile('\s?(city|capital|town|village)\s?', re.I))
if re.search(regCity, l1) and re.match('in|of', c1) and re.match('(?:(?:(?:a|the)\s)?'+ capital +')', l2):
Příklad : Brussels is the main *city of the European Union*.
elif re.search('[Ss]ituat[a-z]*|[Ll]ocate[a-z]*|[Ll]ies|[Bb]orough', string) and re.match ('on|in|by', c1):
Příklad : Paris is *situated in* the centre of France.
elif re.match('[Ss]ituat[a-z]*|[Ll]ocate[a-z]*|[Ll]ies|[Bb]orough', string) and re.match('of', c1):
Příklad : The London *borough of* Islington.
Pravidlo X is capital of Y
if re.search('capital', string) and re.match('of', c1):
Příklad : Barcelona is the *capital of* Catalunia.
Pravidlo X has a population of Y (number = '\d+(?:\.?\,?\d+)*')
if re.search('(?:population|inhabitants|compris)', string) and re.search(number, string):
Příklad : The *population* of Prague is *1.3* million.
Pravidlo X is a centre of Y (predicate = '(?:is|are|were|was|have|has|can|had|[a-z]+s) (?:\s(?:been|be|such as|from))?(?:(?:\s[a-z]{3,})?(?:\s[a-z]+ed))?')
if re.match(predicate, c0) and re.findall('(?:centre|center)', string) and re.match('of', c1):
Příklad : Paris's Champs E'Lysees street is a world *centre of* fashion.
Pravidlo X has a climate Y
if re.findall('climate|weather', l1):
Příklad : Budapest has a mediterranean warm *weather*.
Pravidlo X is called Y
if re.search('(?:name[d]?|called|known)', string) and if re.match('of|as', c1):
Příklad : New York is also *known as* The Big Apple.
Pravidlo X is a city of Y (noun = '[a-z-,\':\"]+')
if re.findall(regCity, l1) and re.match('of', c1) and re.findall(noun, l2):
Příklad : London is a *city of many nationalities*.

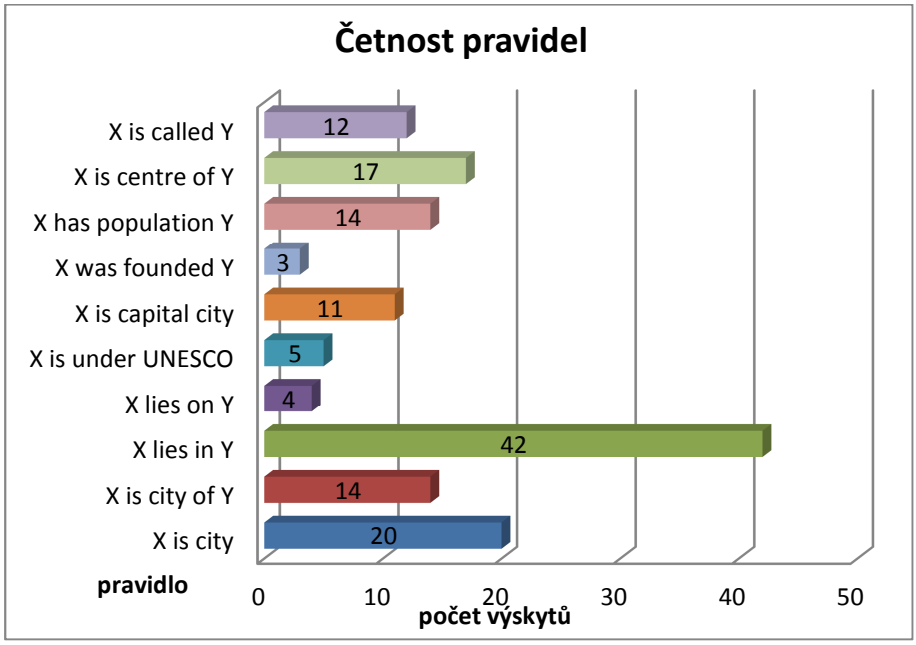
Pravidlo X has been found in Y (noun = '[a-z-,\':\"]+')
if re.search('found|established|formed', string) and re.search(number, string):
Příklad : Humpoletz was *formed* in *1378* by … .

Tabulka 5.1: Výčet nejzákladnějších vytvořených pravidel pro města

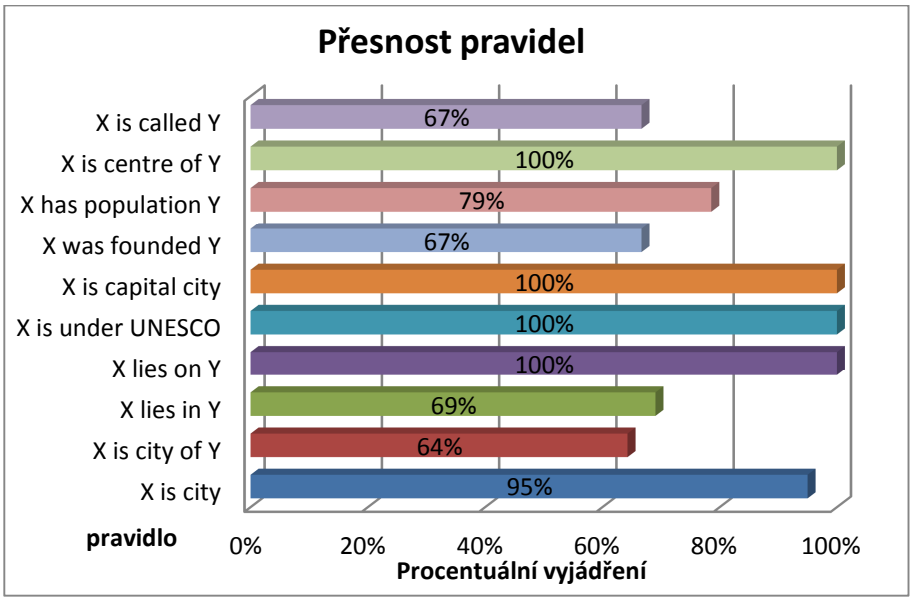
Jednotlivá pravidla byla vyvíjena a validována na vzorku šesti měst : *Prague, Paris, Moscow, Barcelona, Madrid a London.*

Pro samotné experimenty jsme pak vybrali seznam dvaceti míst, mezi kterými se nachází jak velké aglomerace, hlavní města, středně velké i menší městské celky : *Warsaw, Lisbon, Brno, Liverpool, Minsk, Buenos Aires, Atlanta, Los Angeles, Miami, Montreal, Hong Kong, Sydney, Dresden, Bristol, Glasgow, Edinburgh, City of Brussels, Milan, Oslo a Gilford.*

Výsledky experimentů zobrazené v tabulkách :



Obrázek 5.1: Četnost pravidel popisujících město



Obrázek 5.2: Přesnost pravidel popisujících město

Hlavním, třídícím pravidlem je hned to první, uvedené v naší tabulce a selektující téma článku, tedy *X is a city*. Při testování dosáhla jeho úspěšnost určení 95%, v tomto případě uvádíme pojem úspěšnost jako veličinu, udávající v kolika článcích bylo rozeznáno téma (19 z 20 případů). Jediným problémovým prvkem se stal čínský Hong Kong, při pozdější analýze jsme zjistili, že důvodem neúspěchu je samotný článek na Wikipedii, ve kterém je tento termín popsán slovním spojením 'administrative region', jež v našem výčtu validních výrazů nebyl obsažen. Dále zmíněné procentuální údaje již symbolizují přesnost extrakce.

V pořadí druhým, neméně podstatným pravidlem je to, popisující polohu objektu *X lies in/on Y*, kde jsme schopni podle druhu předložky *in* či *on* určit, zdali se jedná o stát či geografický objekt. Předpis se dále skládá ze dvou částí, odlišných podle faktu, zdali je vztah mezi entitami popsán přímo („*The Czech Republic is situated in the centre of Europe.*"), či nepřímo („*London is the capital of the United Kingdom*"). V obou případech jsou využity regulární výrazy, obsahující výčet nejběžněji užívaných termínů. Ve výsledku pak bylo, především díky variantě nepřímo popsaného vztahu, pravidlo velice vytížené, avšak i výrazně neúspěšné. Při experimentech dosáhlo v případě politického situování přesnosti 69 %. Pro názornost uvádíme příklad validní i neúspěšné extrakce : „*Montreal lies in Canada*" vs. „*Montreal lies in in the heart of the city*".

Další námi vytvořená pravidla nebyla tak vytížená a shodu zaznamenala pouze v některých článcích. Větev popisující počet obyvatel našla odpovídající informaci ve 14 případech, z nichž tři uváděly nevalidní znalost, přesnost tohoto pravidla se tak zastavila na bezmála 80 %. Různorodost textů a široká škála forem vypovídajících údaje o populaci, které jsou mnohdy vyjádřeny pouhou zmínkou o populaci a následným holým číslem, zapříčiňují, že dané pravidlo není příliš striktní a může se stát, že daný numerický řetězec symbolizuje jiný údaj než námi požadovaný, například procentuální údaj o vývoji zalidněnosti, jako tomu bylo v onom jednom neúspěšném případě.

Větev, *X is centre of Y*, sice zaznamenala 100% úspěšnost a relativně velký počet shod, avšak toto číslo vyplývá z faktu, že znalost může obsahovat širokou množinu Y entit.

Další dvě pravidla, *X is called Y* a *X is city of Y*, obsahují velice volné parametry, proto se jejich přesnost pohybovala pod úrovní 70 %.

Ve většině zbylých pravidel, ať již těch popsaných v tabulce na předchozí stránce, tak i těch okrajových, jako například *X is under Y, X is tourist destination* došlo sice ke stoprocentní přesnosti, avšak těmto předpisům odpovídající informace se v námi požadovaném formátu vyskytly velmi zřídka.

Příklad kompletního výstupu :

```
./scanner.py –s –t Brno
Brno    is          capital city of South Moravian Region
Brno    is a        city (city)
Brno    is city of          South Moravian Region
Brno    lies in     Czech Republic
Brno    lies in     South Moravian Region
Brno    lies in     southeast of country
Brno    was founded in 1243
```

Obrázek 5.3: Výstup aplikace

5.2.2 Extrakce dat o politicky unifikovaných oblastech

Druhým tématem experimentů se staly světové státy a další druhy územních celků. Rovněž i zde jsou velké odchylky v popisech jednotlivých témat, ať už jsou dány rozdílnou zeměpisnou polohou, ekonomikou či historií. Proto se i v tomto případě specializujeme na základní údaje, které mají všechny státní celky společné. Mezi takové informace patří rozloha, hlavní ekonomická odvětví a rovněž situování či počet obyvatel.

Pravidlo X is a country (regState = re.compile('\s?(kingdom|country|state|republic|province|federal|confederation|commonwealth|nation)\s?', re.I))
if re.search(regState, TOKEN):
Příklad : United *Kingdom* i sone of the most populated states in Europe.
elif re.findall(regState, string):
Příklad : France is a big *state* on the west coast of Europe.
Pravidlo X has natural border with Y (regNature = re.compile('\s?(ocean|sea|mountain|river|lake|gulf)\s?)
if re.search('\s?[Bb](?:order|ound)[a-z]*', string) and re.match('by|with', c1) and re.search (reNature ', l2):
Příklad : Spain is *bordered by* Mediterranean *Sea*.
if re.search('\s?[Bb](?:order|ound)[a-z]*', string) and re.match('in|on|to', c1) and re.search (reNature, l3):
Příklad : Italy is *bordered on* the north by Alps mountains.
Pravidlo X has political border with Y
if re.search('\s?[Bb](?:order|ound)[a-z]*', string) and re.match('by|with', c1) and re.findall ('(?:(?:(?:a|the)\s)?'+ capital +')', l2):
Příklad : Spain has *boundaries with France*.
if re.search('\s?[Bb](?:order|ound)[a-z]*', string) and re.match('in|on|to', c1) and re.findall ('(?:(?:(?:a|the)\s)?'+ capital +')', l3):
Příklad : Luxemburg has *boundaries* in triangle shape with *Belgium*.
Pravidlo X lies in Y (regState = re.compile('\s?(kingdom|country|state|republic|province|federal|confederation|commonwealth|nation)\s?', re.I))
if re.match('in|of', c1) and re.search(regState, l1) and re.findall('(?:(?:(?:a|the))?(?:\s?(?:'+ noun + '|' + capital + '))+)', l2):
Příklad : Czech Republic is a *country in Central Europe*.
if re.match('in|of', c1) and re.search('[Ss]ituat[a-z]*|[Ll]ocate[a-z]*|[Ll]ies', string) and re.findall ('(?:(?:(?:a|the))?(?:\s?(?:'+ noun + '|' + capital + '))+)', l2):
Příklad : Czech Republic is *located in Central Europe*.
Pravidlo X is in organisation Y (regCompany = re.compile('\s?(union|company|organisation|corporation)\s?', re.I))
if re.match('in|of', c1) and re.search(regCompany, l2):
Příklad : France is a founding state *of* the European *Union*.
elif re.findall('(?:\s[Mm]ember\s)', string) and re.match('of', c1):
Příklad : France is a founding *member of* the European Union.

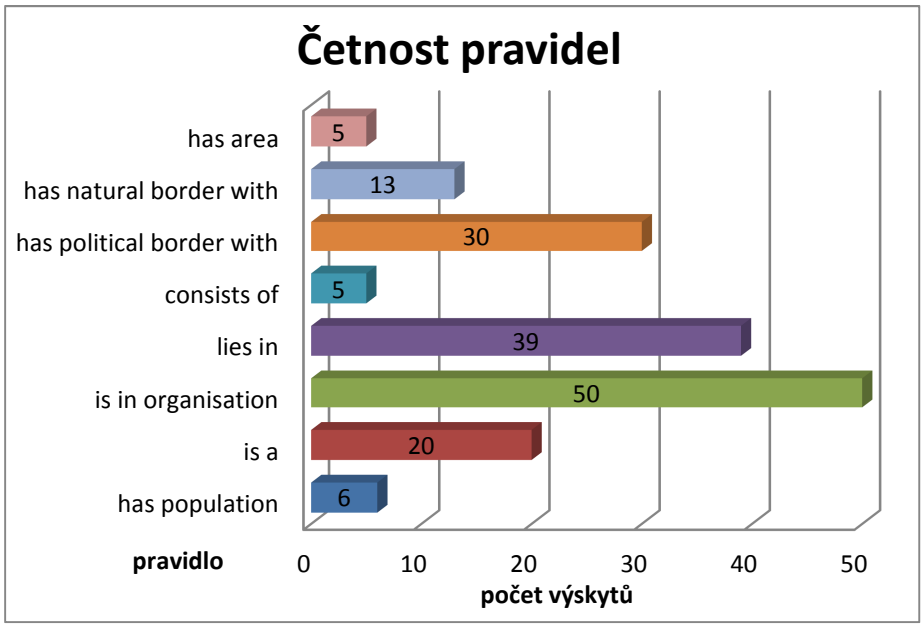
Pravidlo X consists of Y
if re.findall('(?:consist|compris|divided)', string) and re.match('in|into|of', c1) or re.search ('(?:consist|compris|divided)', c0):
Příklad : Czech Republic is *divided into* three substates.
Pravidlo X lies from Y to Z
if re.findall('(?:\s[Ee]xtend|[Ll]ies)', string) and if re.search('from', c0) and re.search('to', c1):
Příklad : Russia *extends from* the Eastern Europe *to* the Eastern Asia.
Pravidlo X has capital city Y
if re.findall('(?:capital)', string) and re.match('is', l1):
Příklad : The *capital* of France *is* Paris.
Pravidlo X has area of Y (number = '\d+(?:\.?\,?\d+)*')
if re.findall(number + '\s?(?:thousand|million|km)?', string) and if re.search('area|territory', string):
Příklad : Czech Republic has an *area* of *78 thousand* square kilometre.
Pravidlo X has area of Y (number = '\d+(?:\.?\,?\d+)*')
if re.findall(number + '\s?(?:thousand|million|km)?', string) and re.search('popul|inhabitant', string):
Příklad : Czech Republic has a *population* of about *10.2 million*.

Tabulka 5.2: Výčet nejzákladnějších vytvořených pravidel pro státy

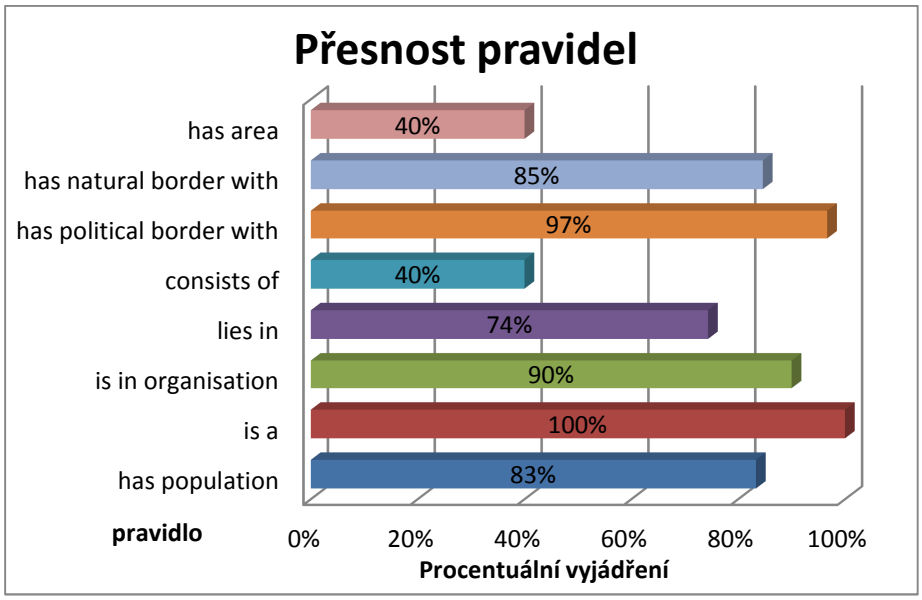
Sběr pravidel probíhal za analýzy těchto územních celků : *United Kingdom*, *France*, *Italy a Germany*.

Pro experimentování s vytvořenými předpisy byla následně vybrána množina 20ti státních celků : *Russia*, *Switzerland*, *Slovakia*, *Poland*, *Brazil*, *Canada*, *Mexico*, *Belgium*, *Netherlands*, *Vietnam*, *Australia*, *Denmark*, *Egypt*, *Portugal*, *Tunisia*, *New Zealand*, *Japan*, *England*, *Scotland a Norway*.

Výsledky experimentů zobrazené v tabulkách :



Obrázek 5.4: Četnost pravidel popisujících stát



Obrázek 5.5: Přesnost pravidel popisujících město

Stejně jako v prvním experimentu, i zde hrálo hlavní roli první pravidlo *X is country*, předzpracovávající základní informaci o daném objektu. Existuje mnoho kategorií politicky rozdělených území, a proto rozlišujeme, zdali se jedná o stát, království či jiný politický druh územní správy. Styl popisu je téměř unifikovaný a výčet validních pojmů, obsažených v pravidle je širší, než tomu bylo v případě městských celků, i proto tak při experimentování došlo ke správnému popisu entity ve 100 % případů.

Stejně tak jsou i v této tématice důležitými popisnými informacemi údaje o situování státu. Příslušný předpis, popisující vztah *X lies in Y*, je složeno ze dvou navzájem nezávislých částí, jedna specializovaná na text, obsahující nepřímo určenou popisnou informaci, například *„Austria is a country in Central Europe“*, druhá pak na větu, obsahující slovesný tvar symbolizující geografickou pozici, jako *„Canada is situated in Northern America“*. První z těchto variant je velice volná, i když vyžaduje, aby entita za přísudkem popisovala územní celek, a tak hlavně ona byla příčinou nepřesného získávání znalostí. V kvantitativním vyjádření přesnost pouze 74 %.

Rovněž i pravidlo určující počet obyvatel území nebylo, co se týká přesnosti, stoprocentně úspěšné, Problém sice nastal pouze v jednom případě, avšak z 20 testovaných celků se tato informace nalezla pouze 6krát. Více o dané situaci v následujícím odstavci.

Podobný případ nastal i u údajů o rozloze a složení státu. Předpisy byly při experimentech shodné s předlohou pouze pětkrát, avšak i přes jejich relativně určitý popis ve třech případech oba přinesly nevalidní informaci a staly se tak nejslabšími články systému. I když jsou i zde konstrukce vysoce specifické, ve výsledku se nalezlo velké množství chyb. V tomto případě přisuzujeme vinu špatnému rozdělení souvětí, pro ukázku *„England consists of the central“*, která postrádá důležitou část informace, a *„England consists of low hills”*, zastupující správně extrahovanou znalost.

Zajímavější a vytíženější oblastí se ukázalo pravidlo, popisující hranice území. V této kategorii navíc rozdělujeme, jedná-li se o lemování hranic politické či přírodní. Údaje se státními hranicemi byly nalezeny 30krát a v součtu přesné z 97 %. Opět na ukázku vypíšeme validní a nevalidní výstup *„Mexico has a border with country Belize“* vs. *„Mexico has a border with country the Gulf“*. Protože pravidlo funguje na principu odchytávání přírodních úkazů a všechny ostatní informace propadají do větve států, byla drobná nepřesnost způsobena chybějícím popisem objektu ve výčtu v regulárním

výrazu. Ani díky právě popsané struktuře implementace, zachytávající pouze přírodní hranice a zbylé objekty řadit jako politické, jsme ani v druhém případě nedostali stoprocentní přesnost, pouze 85%.

Další velice užívanou informací o státech je na Wikipedii údaj o začlenění země do organizací, v našem systému zastoupen vztahem *X is in organization Y*. Díky menší striktnosti specifikace se shodný výraz objevoval ze všech případů nejčastěji a byl účinný z 97 %. *„Canada is in organisation NATO“* představuje správně extrahovanou znalost, *„Denmark is in organisation the Organisation“*, pak informaci irelevantní.

Zbylá pravidla byla buď využita sporadicky či vůbec nikdy. Například díky textům z testovací množiny států byla implementována větev, popisující velikost území tvarem *X lies from Y to Z*, která našla shodu několikrát, avšak ve výsledcích experimentů nebyl zaznamenán ani jeden její případ.

Příklad kompletního výstupu :

```
./scanner.py –s –t Norway
Norway  has political border with       Finland
Norway  has political border with       Russia
Norway  has population  about 4.8 million
Norway  is a    country (Kingdom)
Norway  is in organisation      European Economic Area
Norway  is in organisation      NATO
Norway  is in organisation      OECD
Norway  is in organisation      United Nations
Norway  is in organisation      WTO
Norway  lies in  western portion
Norway  lies in  world
```

Obrázek 5.6: Výstup aplikace

Vypsaný výsledek extrakce obsahuje kromě validních výstupů také dva teoreticky správné výrazy, prakticky však o nich můžeme říci, že jsou špatně, a to Norway lies in Y, v obou případech problém spočívá v užití funkce, která odstřihává text před členem, a tak se zde ta nejdůležitější informace (Northern Europe) ztratila.

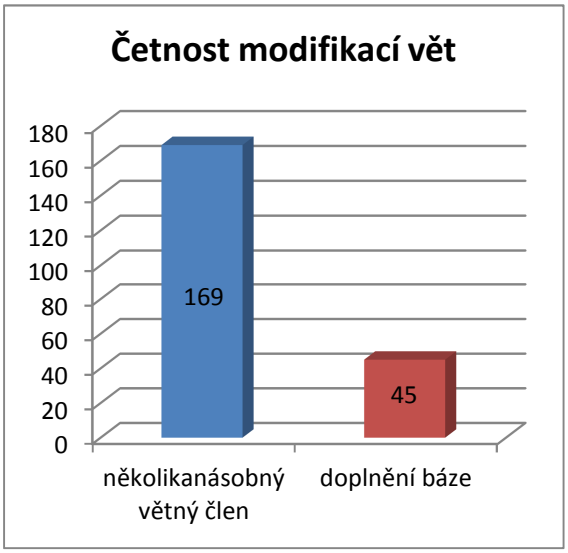
5.3 Korektnost úprav při analýze souvětí

Pro možnosti extrakce jsme museli nejprve vstupní text prozkoumat a v případě souvětí jej rozdělit a následně správně modifikovat tak, abychom pokud možno neztratili důležité informace a naopak získali věty nové, které splňují kritéria pro další rozbor. Podrobnější informace se nachází v kapitole 4.3.5.

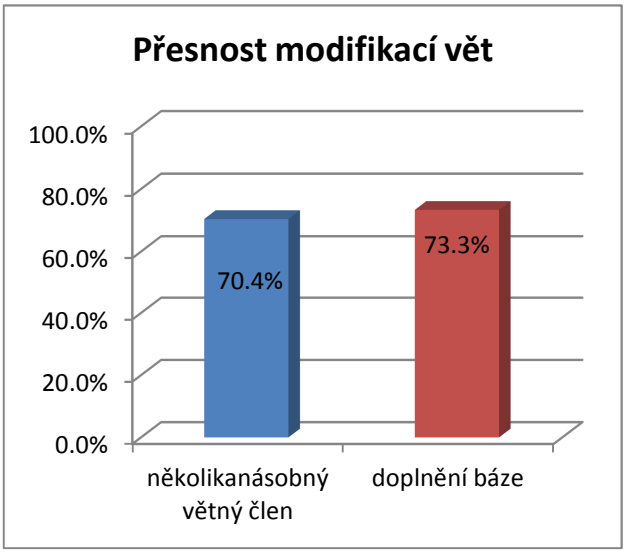
Jelikož je všeobecně analýza jazyka velice náročné odvětví, nejsou naše řešení několikanásobných větných členů, vět vedlejších, vložených a dalších problematik, stoprocentně úspěšné. Pro získání konkrétních výsledků jsme naše modifikační funkce podrobili zkoušce a při experimentu s městskými oblastmi, popsaném v podkapitole 5.2.1 jsme zkoumali jejich výstupy.

Upravovány byly pouze souvětí, které v sobě na pozici podmětu ve větě hlavní obsahovaly hledaný token, proto se očekávala relativně malá četnost korektur, i když se v počátku jednalo o velké množství vstupních dat.

Výsledné grafy experimentů s funkcemi analyzujícími věty.



Obrázek 5.7: Četnost vytížení fukncí



Obrázek 5.8:Úspěšnost funkcí

Experiment ukázal, že se ve vstupních textech vyskytovaly převážně souvětí obsahující výčty informací a přesnost úprav se v obou funkcích pohybovala kolem 70%.

Jako validní byly považovány věty, které neobsahovaly gramatické chyby, jako například dvě spojky vedle sebe, a věty, které dávaly správný věcný význam.

Pro názornost uvedeme ukázky z obou variant :

Wroclaw is situated on the River Oder.
(ze souvětí : Wroclaw is the chief city in south-western Poland, situated on the River Oder)
Dresden is situated in a valley on the Czech border.
Bristol was granted County status in 1373.

Obrázek 5.9: Příklady správně upravených vět

Sydney has a reputation as arts.
(ze souvětí : Sydney has a reputation as an international centre for commerce, arts,..)
Oslo has been listed as Paris.

Obrázek 5.10: Příklady špatně upravených vět

Jak již bylo zmíněno, analýza jazyka je vysoce náročná disciplína a tak nejsou výsledné hodnoty příliš překvapující. Hlavní příčiny neúspěchu jsou popsány v následující podkapitole.

5.4 Diskuse k problémům a nedostatkům

Při validaci a zkoumání jednotlivých funkčních částí kódu jsme museli často čelit vzniklým problémům, mnohdy se nám je podařilo odstranit a v některých případech jsme na jejich fixaci byli buď krátcí, nebo nám je náš způsob implementace nedovoloval řešit. I díky těmto nedostatkům a chybám se pravidla v experimentech mnohdy potýkala s nízkým procentem přesnosti.

Při prvotních rozborech anglických textů jsme došli k závěru, že nejdůležitější informace se v drtivé většině případů vyskytují v přední části věty a tak ji v našem systému zkoumáme pouze do určité hloubky a zbytek textu zahazujeme. V pozdějším testování jsme však narazili na věty jako je tato „*Brussels has grown from a 10th-century fortress town founded by a descendant of Charlemagne into a metropolis of more than 1 million inhabitants.*“, kde je informace o počtu obyvatel uvedena až na jejím samotném chvostu.

Nedostatkem, který má poměrně velký dopad na množství extrahovaných znalostí, se jeví vlastnost zkoumání vět, které v sobě po rozboru a úpravách obsahují jako podmět hledaný výraz. Urychlujeme tím sice samotný proces extrakce a vyhýbáme se větším informačním nepřesnostem, avšak ve článcích se často popisuje hlavní objekt jeho nadřazenou třídou. Pro lepší pochopení: „*Burj Khalifa is the highest building on the Earth. The skyscraper is situated in Dubai…”*, druhá věta se tedy neanalyzuje, což má neblahý vliv na počet extrahovaných znalostí. Modifikace systému by byla bohužel náročná a komplexní.

Jak již bylo několikrát zmíněno, problematika dělení souvětí je velice komplikovaná a tak i účinnost námi navrženého jednoduchého analyzátoru není stoprocentní, například při souvětí „*ABBA was a Swedish pop music group formed in Stockholm in 1972, consisting of Anni-Frid Lyngstad*“ se nám po úpravě dostane nesprávného výsledku. Především tento faktor hrál hlavní roli v neúspěšných modifikacích vět, zkoumaných v předešlém experimentu.

Toto všechno jsou nedostatky, kterých by se změnou návrhu a komplexnějším řešením dalo vyvarovat, některé měly ovšem složitější původ. Například i anotační aplikace Tree-Tagger není vždy 100% úspěšná, a pro ukázku ve větě „*Italy borders with France*“ značkuje program chybně sloveso *borders* jako množné číslo od podstatného jména *border*. Tento důvod zapříčiňuje, že větě chybí přísudek, je systémem zahozena a my ztrácíme důležitou informaci. Některé věty se již do naší aplikace dostaly poškozené, jako například „*Norway has a total area of and a population of about 4.8 million*“.

Posledním bodem, který stojí za zmínku, je informační neúplnost vět, se kterou jsme se často setkali. Pravidla našeho systému vycházejí ze vztahu *X is Y*, kde prvky X a Y představují jmenné fráze, v anglických textech se druhá z podmínek mnohdy vypouští, jako ve větě „*The apple is the most widely known of the many member of genus Malus*“, ve které chybí věcná informace o tom, co ono jablko v podstatě je.

5.5 Vyhodnocení extrakce

Pravidla byla vyvíjena na širší validační sadě článků a většina z nich dosahovala velmi vysokého procenta úspěšnosti v otázce přesnosti, proto se jejich šance na efektivní extrakci při testování zdála být vysoká. Přesto se v následných experimentech tyto předpoklady ukázaly jako mylné a do jisté míry příliš ambiciózní.

Některé předpisy, které měly oba zkoumané obory společné, byly sestavovány pokaždé jinak. V určitých pravidlech to bylo zapříčiněno jinou konstrukcí vět, avšak u některých informací, jako například u počtu obyvatel či situování hledaného objektu, byly sestaveny v obou případech odlišně z důvodu srovnání, který z vytvořených předpisů bude efektivnější. Pro ukázku v otázce obyvatel v prvním okruhu dostáváme nižší přesnost než v druhém, zde se však údaj o populaci objevuje s nižší četností a má tedy horší pokrytí. Přestože nevíme, zdali se opravdu hledaná informace vyskytovala v textech pouze několikrát, můžeme tvrdit, že zkombinováním obou metod získáme vyšší přesnost.

V naší podobě systému závisí primární úspěšnost splnění úlohy na správném rozpoznání tématu při předzpracování, nacházejícím se v úvodní části pravidel. Pro podrobnější extrakci znalostí z textu je tato část klíčová a její správná funkčnost, zajišťující analýzu zbylého textu, závisí na výčtu podtémat, vložených do regulárního výrazu příslušného pravidla. Pro příklad kategorie měst v sobě zahrnuje v systému termíny *city, capital, town a village*, a čím více budeme slovník validních výrazů rozšiřovat, tím se bude zvyšovat pravděpodobnost úspěšného zařazení a následné extrakce.

6 Závěr

6.1 Zhodnocení práce

Cílem bakalářské práce bylo seznámení s problematikou extrakce informací z textu a následné navržení a implementování systému, obsahujícího automatické extrakční vzory, užité pro sběr znalostí z článků informačního serveru Wikipedie, a na závěr pak podrobení vytvořené aplikace sérií experimentů.

Vstupní neformátovaný text jsme napřed podrobili morfologické analýze a následně posloupností procesů modifikovali a rozdělili do funkčních větných bloků. V této sekci jsme se snažili řešit problematiky jako například nevyjádřené podměty a přísudky, užívání zájmen či nepřímých řečí. Dále jsme již takto zpracované věty podrobili samotné extrakci informací za užití pravidlových vzorů.

V naší podobě systému závisí úspěšnost splnění úlohy na správném rozpoznání tématu při předzpracování, nacházejícím se v úvodní části pravidel. Pro podrobnější extrakci znalostí z textu je tato část klíčová a její správná funkčnost, zajišťující analýzu zbylého textu, závisí na obsáhlosti podtémat vložených do regulárního výrazu příslušného pravidla. Odstraněním takového postupu ve spojení s obecněji napsanými pravidly, skládajícími se pouze z regulárních výrazů, bychom získali na jednu stranu systematičtější řešení problematiky extrakce, avšak po nabytých zkušenostech s různorodostí stylu psaní článků a složitostí vět, jsme usoudili, že taková implementace, splňující aspoň do jisté míry kritérium přesnosti, by byla obtížná. Proto jsme se rozhodli pro tento kombinovaný postup, využívající možností regulárních výrazů pouze z části a užívající skupin pravidel pro každou kategorii zvlášť. Tímto způsobem bychom mohli neustále vyvíjet a rozšiřovat pole předpisů, pro názornost a potřebu experimentů jsme implementovali pouze některá.

Již při validaci pravidel jsme zjistili, že lepší návrh aplikace ve fázi zpracování a analýze větných celků by vedl k výrazně efektivnější extrakci.

Experimenty pak prokázaly, že i když je databáze Wikipedie obrovským potencionálním zdrojem znalostí, strukturovaných například do databází, jejich analýza a samotný proces extrakce jsou vysoce složitým a komplexním problémem, na který jsou potřeba jemné, avšak rozsáhlé extrakční vzory, a jež se nám podařilo úspěšně implementovat pouze z části.

Z výčtu kritérií, jež byla od aplikace požadována, jsme dosáhli za ideálních podmínek rychlosti extrakce a u některých pravidel i přesnosti. Akce okamžitého vyhledání zadaného článku, následné analýzy a extrakce, proběhly ve všech případech během několika vteřin.

6.2 Možná rozšíření

Díky užití pomocného skriptu, který zabalené články z XML databáze vyselektoval a předložil naší aplikaci víceméně již holý text, lze náš program užít i na jiná, libovolná textová data, nikoliv pouze články Wikipedie.

Literatura

[1] *Wiki*, [online], [cit. 2010-05-16].
URL < http://cs.wikipedia.org/wiki/Wiki >

[2] SARAWAGI, S: *Information Extraction,* Now Publishers.Inc, 2008,
ISBN 978-1-6019-8188-2.

[3] MOENS, M.: *Information Extraction: Algorithms and Prospects in a Retrieval Context.*
Springer, 2006, ISBN 978-1-4020-4987-3.

[4] *Developing Language Processing Components with GATE,* [online], [cit. 2010-05-16].
URL < http://gate.ac.uk/sale/tao/#x1-2090008 >

[5] *Základy jazyka RDF*, Doc. Ing. Vojtěch Svátek, Dr. [online], [cit. 2010-05-16].
URL < http://nb.vse.cz/~svatek/rzzw/RDF_l11.pdf >

Seznam příloh

- Příloha A: Obsah přiloženého DVD

Obsah přiloženého DVD

Adresářová struktura:

/bakalarska prace	bakalářské práce
bakalarska_prace.doc	dokument ve zdrojové podobě
bakalarska_prace.pdf	práce v PDF formátu
/program	
/database	databáze Wikipedie
/lists	seznamy objektů
/temp	pomocná složka pro extrakci
/utility	použité nástroje
/treeTagger	morfologický analyzátor Tree-Tagger
output.txt	soubor s výsledky experimentů
testscript.py	script pro testování
wikiextractor.py	extrakční aplikace
README.txt	popis spuštění skriptů